PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-272847

(43)Date of publication of application : 08.10.1999

(51)Int.Cl. G06T 1/00

H04N 11/04

(21)Application number : 10-074719

(22)Date of filing : 23.03.1998

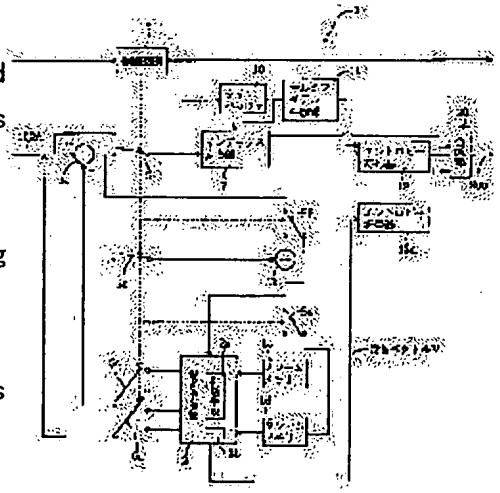
(71)Applicant : SEIKO EPSON CORP

(72)Inventor : HOSHINA SHOJI

(54) DEVICE AND METHOD FOR ENCODING MOVING IMAGE CONSISTING OF MULTICOLOR IMAGE, AND DEVICE AND METHOD FOR DECODING THE SAME MOVING IMAGE

(57)Abstract:

PROBLEM TO BE SOLVED: To apply motion compensation even when a moving image consisting of multicolor images is encoded and decoded and to attain the conflicting purposes of making a strict identification of a block at the time of motion compensation and of preventing an increase in the number of moving vectors.



SOLUTION: An encoding device 1 for moving image consisting of multicolor images performs motion compensation when generating a moving image. A judging means 2a calculates the number of different pixels at corresponding positions of both blocks and judges that the blocks are the same or similar when the number of different pixels is less than a specific value to generate a moving vector V. The judging means 2a also judges whether or not the moving vector V is used by divided block, and judges that the

blocks are the same or similar when the number of differences between both the blocks is less than a specified number to perform motion compensation by using the moving vector V of the macroblocks. Reverse algorithm is used for decoding.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－272847

(43)公開日　平成11年(1999)10月８日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｔ　1/00		Ｇ０６Ｆ　15/66	３３０Ｐ
Ｈ０４Ｎ　11/04		Ｈ０４Ｎ　11/04	Ｚ
		Ｇ０６Ｆ　15/66	３１０

審査請求　未請求　請求項の数17　ＯＬ　（全 41 頁）

(21)出願番号　特願平10－74719

(22)出願日　平成10年(1998)３月23日

(71)出願人　000002369
セイコーエプソン株式会社
東京都新宿区西新宿２丁目４番１号

(72)発明者　保科　彰治
長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社内

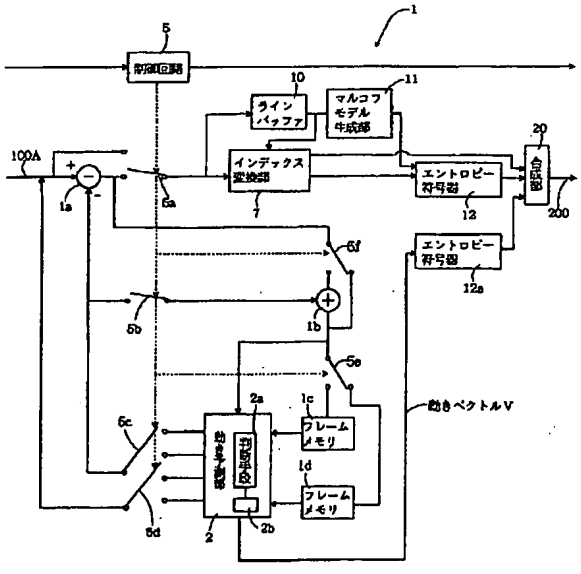
(74)代理人　弁理士　鈴木　喜三郎　（外２名）

(54)【発明の名称】　マルチカラー画像からなる動画像の符号化装置およびその方法並びにマルチカラー画像からなる動画像の復号化装置およびその方法

(57)【要約】

【課題】　マルチカラー画像からなる動画像の符号化や復号化を行う際にも動き補償を適用できるようにする。また、動き補償の際のブロックの同一性の判断を厳しくしたいが、動きベクトルの数を増やしたくないという相反する目的を同時に達成する。

【解決手段】　このマルチカラー画像からなる動画像の符号化装置１は、動画像を形成する際、動き補償を行う。そして、判断手段２ａによって両ブロックの対応する位置の画素が異なる場合の数を算出し、その不一致数が所定数以下のときには同一または類似のブロックと判断し、動きベクトルＶを作成している。また、判断手段２ａは、各分割されたブロック毎に動きベクトルＶを用いるか否かの判断を行い、両ブロックの不一致数が所定数以下のときに同一または類似のブロックと判断し、マクロブロックの動きベクトルＶを利用して動き補償を行うようにしている。なお、復号化の際は逆となるアルゴリズムを使用している。



【特許請求の範囲】

【請求項１】　インデックスコードが付与されたカラー画素データからなる静止画のフレームを連続させて動画像を形成する際、符号化対象の着目フレームをそれぞれＮ×Ｎピクセル（Ｎは４以上の整数）からなる複数のブロックに分割し、その着目フレーム中のブロックが参照フレーム中のブロックと同一または類似しているとき、着目フレーム中のブロック位置と参照フレーム中のブロック位置との相対変位を動きベクトルとして動き補償を行うマルチカラー画像からなる動画像の符号化装置において、上記動きベクトルを用いるか否かの判断手段を設け、この判断手段によって両ブロックの対応する位置の画素が異なる場合の数を算出し、その不一致数が所定数以下のときには同一または類似のブロックと判断し、動きベクトルを作成することを特徴とするマルチカラー画像からなる動画像の符号化装置。

【請求項２】　前記ブロックは、ＭＮ×ＭＮピクセル（Ｍは２以上の整数）からなるより大きなマクロブロックを小ブロック化手段によってＭ×Ｍに分割され、前記判断手段は、各分割されたブロック毎に前記動きベクトルを用いるか否かの判断を行うために、前記参照フレーム内の対応するブロックの対応する位置にある画素と当該ブロックの画素とが異なる場合の数を算出し、その不一致数が所定数以下のときに同一または類似のブロックと判断し、上記マクロブロックの動きベクトルを利用して前記動き補償を行うようにしたことを特徴とする請求項１記載のマルチカラー画像からなる動画像の符号化装置。

【請求項３】　前記ブロックを８×８ピクセルからなるブロックとし、前記所定数を０～５の整数値のいずれか１つとしたことを特徴とする請求項１または２記載のマルチカラー画像からなる動画像の符号化装置。

【請求項４】　前記動きベクトルを用いないと判断したブロックを、前記着目フレームの符号化時の走査方向と直角となる方向に拾っていくと共に、着目画素の囲りの参照画素を利用して当該各ブロック内の各画素を符号化することを特徴とする請求項１、２または３記載のマルチカラー画像からなる動画像の符号化装置。

【請求項５】　インデックスコードが付与されたカラー画素データからなる静止画のフレームを連続させて動画像を形成する際、符号化対象の着目フレームをそれぞれＮ×Ｎピクセル（Ｎは４以上の整数）からなる複数のブロックに分割し、その着目フレーム中のブロックが参照フレーム中のブロックと同一または類似しているとき、着目フレーム中のブロック位置と参照フレーム中のブロック位置との相対変位を動きベクトルとして動き補償を行うマルチカラー画像からなる動画像の符号化方法において、上記動きベクトルを用いるか否かの判断工程を設け、この判断工程によって両ブロックの対応する位置の画素が異なる場合の数を算出し、その不一致数が所定数以下のときには同一または類似のブロックと判断することを特徴とするマルチカラー画像からなる動画像の符号化方法。

【請求項６】　前記ブロックは、ＭＮ×ＭＮピクセル（Ｍは２以上の整数）からなるより大きなマクロブロックを小ブロック化工程によってＭ×Ｍの数に分割され、前記判断工程は、各分割されたブロック毎に前記動きベクトルを用いるか否かの判断を行うために、前記参照フレーム内の対応するブロックの対応する位置にある画素と当該ブロックの画素とが異なる場合の数を算出し、その不一致数が所定数以下のときに同一または類似のブロックと判断し、上記マクロブロックの動きベクトルを利用して前記動き補償を行うようにしたことを特徴とする請求項５記載のマルチカラー画像からなる動画像の符号化方法。

【請求項７】　前記ブロックは、ＭＮ×ＭＮピクセル（Ｍは２以上の整数）からなるより大きなマクロブロックを小ブロック化工程によってＭ×Ｍの数に分割され、前記判断工程は、各分割されたブロック毎に前記動きベクトルを用いるか否かの判断を行うために、前記参照フレーム内の対応するブロックの対応する位置にある画素と当該ブロックの画素とが異なる場合の数を算出し、その不一致数が所定数以下のときにＮ×Ｎピクセルのブロックについて同一または類似のブロックと判断し、上記マクロブロックの動きベクトルを利用して前記動き補償を行うようにしたことを特徴とする請求項５記載のマルチカラー画像からなる動画像の符号化方法。

【請求項８】　前記ブロックを８×８ピクセルからなるブロックとし、前記所定数を０～５の整数値のいずれか１つとしたことを特徴とする請求項５、６または７記載のマルチカラー画像からなる動画像の符号化方法。

【請求項９】　前記動きベクトルを用いないと判断したブロックを、前記着目フレームの符号化時の走査方向と直角となる方向に拾っていくと共に、前記着目画素の囲りの参照画素を利用して当該各ブロック内の各画素を符号化することを特徴とする請求項５，６，７または８記載のマルチカラー画像からなる動画像の符号化方法。

【請求項１０】　入力される対象符号化データをインデックスコードが付与されたカラー画素データに復号するエントロピー復号化手段と、該カラー画素データからなる静止画を連続させて動画像を形成する動画像形成手段とを備え、動画像形成手段は、復号化対象の着目フレームをそれぞれＮ×Ｎピクセル（Ｎは４以上の整数）からなる複数のブロックに分割し、その着目フレーム中のブロックに関する復号化データ中にそのブロックの相対変位の方向を示す動きベクトルが存在するとき、参照フレーム中の同一または類似のブロックを利用してそのブロックを復号する動き補償を行うマルチカラー画像からなる動画像の復号化装置において、上記動きベクトルは、上記着目フレーム中のブロックと上記参照フレーム中の

対応するブロックの対応する位置の画素が異なる場合のその不一致数が所定数以下のときに発生させられたものであることを特徴とするマルチカラー画像からなる動画像の復号化装置。

【請求項１１】　前記ブロックを、ＭＮ×ＭＮピクセル（Ｍは２以上の整数）からなるより大きなマクロブロックから小ブロック化手段によってＭ×Ｍの数に分割し、上記マクロブロックの動きベクトルを利用して上記小さな各ブロックを復号するか否か判断する判断手段を有していることを特徴とする請求項１０記載のマルチカラー画像からなる動画像の復号化装置。

【請求項１２】　前記ブロックを、８×８ピクセルからなるブロックとし、前記所定数を０～５の整数値のいずれか１つとしたことを特徴とする請求項１０または１１記載のマルチカラー画像からなる動画像の復号化装置。

【請求項１３】　前記動きベクトルを有しないブロックを、前記着目フレームの符号化時の走査方向と直角となる方向につなぎ合わせて復号すると共にその復号の際、着目画素の囲りの参照画素を利用して当該動きベクトルを有さないブロック内の各画素を復号化することを特徴とする請求項１０、１１または１２記載のマルチカラー画像からなる動画像の復号化装置。

【請求項１４】　入力される対象符号化データをインデックスコードが付与されたカラー画素データに復号するエントロピー復号化工程と、該カラー画素データからなる静止画を連続させて動画像を形成する動画像形成工程とを備え、動画像形成工程は、復号化対象の着目フレームをそれぞれＮ×Ｎピクセル（Ｎは４以上の整数）からなる複数のブロックに分割し、その着目フレーム中のブロックに関する復号化データ中にそのブロックの相対変位の方向を示す動きベクトルが存在するとき、参照フレーム中の同一または類似のブロックを利用してそのブロックを復号する動き補償を行うマルチカラー画像からなる動画像の復号化方法において、上記動きベクトルは、上記着目フレーム中のブロックと上記参照フレーム中の対応するブロックの対応する位置の画素が異なる場合のその不一致数が所定数以下のときに発生させられたものであることを特徴とするマルチカラー画像からなる動画像の復号化方法。

【請求項１５】　前記ブロックを、ＭＮ×ＭＮピクセル（Ｍは２以上の整数）からなるより大きなマクロブロックから小ブロック化工程によってＭ×Ｍの数に分割し、上記マクロブロックの動きベクトルを利用して上記小さな各ブロック復号するか否か判断する判断工程を有していることを特徴とする請求項１４記載のマルチカラー画像からなる動画像の復号化方法。

【請求項１６】　前記ブロックを、８×８ピクセルからなるブロックとし、前記所定数を０～５の整数値のいずれか１つとしたことを特徴とする請求項１４または１５記載のマルチカラー画像からなる動画像の復号化方法。

【請求項１７】　前記動きベクトルを有しないブロックを、前記着目フレームの符号化時の走査方向と直角となる方向につなぎ合わせて復号すると共にその復号の際、着目画素の囲りの参照画素を利用して当該動きベクトルを有さないブロック内の各画素を復号化することを特徴とする請求項１４、１５または１６記載のマルチカラー画像からなる動画像の復号化方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、マルチカラー画像からなる動画像の符号化装置およびその方法ならびにマルチカラー画像からなる動画像の復号化装置およびその方法に関する。そして、さらに詳細に述べれば、マルチカラー画像を符号化および復号化する際の動き補償の改良に関する。

【０００２】

【従来の技術】従来からパソコンやゲーム機器等では、マルチカラー画像と呼ばれる画像が使用されている。このマルチカラー画像とは、代表色画像とか限定色画像等とも呼ばれているもので、図４１に示すように、特定の色、すなわち特定のＲ（赤）、Ｇ（緑）、Ｂ（青）の値を持つ色に対してインデックスを付与し、そのインデックスのデータを利用して、１６色や２５６色等の限定された代表色で表現するようにした画像のことである。

【０００３】このようなマルチカラー画像のデータは、仮にＲ、Ｇ、Ｂの各色が８ビット（２５６種）で表されるとしたら、合計２４ビット必要になるのであるが、インデックスそのものも例えば８ビットで表示するようにしているので、相当な圧縮率となっている。しかし、圧縮はされているが、それでも情報量が多いため、何の工夫もせず、そのままの形で処理すると、メモリ容量が大きくなり、また通信速度も遅くなり実用的でない。したがって、マルチカラー画像も他の画像データと同様にその圧縮技術は極めて重要なものとなる。特に、マルチカラー画像は、その色の数が限定されていることから、ロスレスでの符号化および復号化、すなわち可逆的な圧縮技術が必要とされている。

【０００４】一方、近年、データ圧縮の手法の一つとして、エントロピー符号器および復号器を用いた技術が注目されている。このエントロピー符号化および復号化技術の一つとして、例えば、算術符号化および復号化の技術を用いたものがある。特開昭６２－１８５４１３号公報、特開昭６３－７４３２４号公報、特開昭６３－７６５２５号公報等に記載されている。

【０００５】図３５に、このような技術を用いた従来のマルチカラー画像の符号化システム５０および復号化システム６０を示す。この符号化システム５０は、ラインバッファ５１と、エントロピー符号器５２とを含むものである。入力されるインデックスのデータ、すなわちカラー画素データ１００Ａは、ラインバッファ５１および

エントロピー符号器５２へ入力される。このカラー画素データ１００Ａは、図３６に示すように、いずれもラスタースキャンされ水平走査順に順次画素データとして入力される。なお、このインデックスのデータ、すなわちカラー画素データ１００Ａを作成する方法としては、入力する色の順番にインデックスを付与する方法が一般的であり、図４１に示すように、インデックスの番号が近いもの、例えば「1」と「2」でもその色が大きく異なったり、インデックスの番号が遠いもの、例えば「１００」と「２００」でもその色は近似している現象が生じている。このような現象を避けるため、特開平５－３２８１４２に示されるように、色の近いものに連続番号を付与するようにしたものも現れている。

【０００６】符号化システム５０中のラインバッファ５１は、参照画素生成手段として、既に入力されたカラー画素データ１００Ａから、符号化対象画素Ｘに対する参照画素データＡ，Ｂ，Ｃ，Ｄを作成する。すなわち、ラインバッファ５１は、画像をスキャンするときにｎライン（１～５ライン程度が多い）分の履歴を記憶しておく。そして、符号化対象画素Ｘのカラー画素データ１００Ａが入力されるごとに、この直前の画素Ａと、周辺の画素Ｂ，Ｃ，Ｄとからなる一連の画素データを参照画素データ１１０としてエントロピー符号器５２へ向けて出力する。

【０００７】このエントロピー符号器５２は、例えば、算術符号化またはハフマン符号化などの手法を用いて形成される。そして、参照画素データ１１０を状態信号として用い、対象カラー画素データ１００Ａを符号化データ２００に変換出力する。

【０００８】一方、復号化システム６０は、ラインバッファ６１とエントロピー復号器６２を含んで構成される。ここにおいて、ラインバッファ６１とエントロピー復号器６２は、入力される符号化データ２００を符号化システム５０のラインバッファ５１、エントロピー符号器５２とは全く逆の手順で復号化出力するように形成されている。

【０００９】このようにして、符号化システム５０と、復号化システム６０とは、互いに全く逆のアルゴリズムを用いて、カラー画素データ１００Ａを符号化データ２００に符号化し、さらにこの符号化データ２００をカラー画素データ１００Ｂに復号化して出力することができる。したがって、このシステムは、各種用途に幅広く用いることができる。

【００１０】ところで、このようなシステムでは、カラー画素データ１００Ａの値、すなわちインデックスの番号が一定の番号付近に偏るとそのデータの圧縮率が向上する。また、このシステムでは、参照画素データ１１０を、エントロピー符号器５２、エントロピー復号器６２の状態信号として使用している。したがって、その状態数、すなわち、参照画素数を多くとれば、同様にデータ圧縮率は向上する。すなわち、算術符号化またはハフマン符号化などの手法を用いてエントロピー符号器５２および復号器６２を構成する場合には、０または１のシンボルの発生確率に大きな偏りがあると、データの圧縮率を高めることができる。これは、エントロピー符号化技術では、発生確率の高い入力データには短い符号化データを割り当て、発生率の低い入力データには相対的に長い符号化データを割り当てるからである。

【００１１】シンボル、すなわちインデックスの番号の発生確率の大きな偏りを得るために、従来より、入力データをいくつかの状態に分類して符号化することが行われている。なぜなら、分類をしないと、良い圧縮率は得られないからである。例えば、図３５に示す従来の手法では、ラインバッファ５１，６１を用い、参照画素データを作成し、これを分類用の状態信号としてエントロピー符号器５２およびエントロピー復号器６２へ入力している。そして、これらエントロピー符号器５２およびエントロピー復号器６２は、前記状態信号を用いることにより入力データを分類し、符号化および復号化を行っている。すなわち、参照画素データの各状態の発生確率を計算し、その発生確率の高い組み合わせのものに短い符号化データを割り当てている。そして、これによりデータの圧縮率を高めている。

【００１２】しかし、前述したエントロピー符号器５２およびエントロピー復号器６２では、参照画素データの状態数に対応した数の符号化パラメータテーブルが必要となる。このため、圧縮率を高めるために参照画素数を大きくとればとるほど、符号化および復号化のパラメータテーブルが大きくなる。このため、エントロピー符号器５２およびエントロピー復号器６２が大型化かつ高価となってしまうという問題がある。

【００１３】例えば、カラー画素データ、すなわちインデックスの番号を４ビットデータ（１６種）で構成し、しかも参照画素データ１１０の画素数が４である場合を想定する。この場合には、符号化および復号化パラメータテーブルの状態数は、４画素×４ビット＝１６ビット分の状態数、すなわち2^{16}の状態数をとる。このため、2^{16}＝６５５３６通りのパラメータテーブルを用意しなければならない。このことからも、参照画素を１つ増やすごとに、その分、符号化および復号化パラメータテーブルが極めて大きくなり、エントロピー符号器５２およびエントロピー復号器６２を構成するハードウェアが大型化してしまうことが理解される。しかも、対象画素も４ビット、すなわち、４プレーンで構成され、各プレーンに１ビットずつの信号が割り当てられ、結果として４ビットで１６通りの値（色）をとるので、パラメータテーブルは６５５３６×１６の大きさを持つテーブルとなる（図３７参照）。

【００１４】このような問題に対し、対象画素のカラーシンボル、すなわち色に対応するインデックスの番号の

出現頻度の偏りを計算し、出現頻度順位に対応して、インデックスの番号を並び替える色順位変換の方法（特開平６－２７６０４１号）がある。すなわち、これにより出現頻度の高いものに短い符号化データを割り当て、さらに、圧縮率を高めている。また、この公開公報には、エントロピー符号器５２およびエントロピー復号器６２の中に縮退した状態数に応じてパラメータテーブルを小さくさせる技術も開示されている。

【００１５】この特開平６－２７６０４１号公報に示されている状態数を縮退するシステムの特徴は、図３８に示すように、従来の符号化システム５０や復号化システム６０と同様にエントロピー符号器５２およびエントロピー復号器６２に参照画素データ１１０を状態信号として入力するわけであるが、その入力に際し、その状態信号１４０を、ラインバッファ５１，６１から出力される参照画素データ１１０を縮退する状態縮退器５３，６３によって生成する点にある。

【００１６】この状態縮退器５３、６３は、入力される参照画素データ１１０を、より少ないビット数の状態信号１４０に縮退し、対応するエントロピー符号器５２およびエントロピー復号器６２へ向け出力するように構成されている。なお、予測器５４，６４は、それぞれカラーシンボルの出現頻度に基づいてカラー画素データを色順位に変換するためおよびその逆を行うための色順位テーブル（詳細は後述）をそのメモリーに保有しているものである。

【００１７】なお、縮退とは、縮退後の状態数に、元の状態を分類する操作である。この分類は、分類後のエントロピー（１つのシンボルを表示するための平均情報量）が最少となるように、その組み合わせを選択して行う。そして、縮退後の状態数、すなわち、分類された後の状態数に対して識別ビットを付加する。これが状態信号１４０である。

【００１８】ところで、状態縮退器５３，６３に用いる縮退テーブルとしては、参照画素データ１１０のカラーシンボルの組み合わせパターンと、縮退データとの関係を特定する縮退テーブルを設定し、この縮退テーブルを用い、入力される参照画素データ１１０のカラーシンボルの組み合わせパターンを、縮退データに変換出力する方法がある。

【００１９】図３９には、このような手法を用いて行われる縮退動作の一例が示されている。ここでは、説明を簡単にするために、図３９（Ａ）に示すよう、符号化対象画素Ｘに対し、Ａ，Ｂ，Ｃの３つの画素から形成されるマルコフモデルを参照画素パターンとして用いる場合を例にとり説明する。

【００２０】参照画素が、図３９（Ａ）に示すように、３つの画素から構成される場合には、そのカラーシンボルの組み合わせパターンは、図３９（Ｂ）に示すように５通りとなる。すなわち、３つに画素のカラーシンボルが全て一致するパターンと、２つのカラーシンボルのみが一致する場合に該当する３つのパターンと、全ての画素のカラーシンボルが異なるパターンの計５つのパターンに分類される。

【００２１】したがって、図３９（Ｂ）に示すテーブルを状態縮退器５３，６３の縮退テーブルとして用いることにより、本来３つの画素の組み合わせが取りうる2^{12}パターンの状態を、図３９（Ｂ）に示す５つの状態Ｓ１～Ｓ５に縮退することができる。このようにすることによって、参照画素データ１１０を効果的に縮退し、エントロピー符号器５２およびエントロピー復号器６２の状態数を大幅に少なくすることができる。

【００２２】ところで、このような算術符号化および復号化の一般的な手法は、既に１画像符号化標準ＪＢＩＧ（インターナショナルスタンダードＩＳＯ／ＩＥＣ１１５４４）のｐ２６～４４およびｐ４４～ｐ５０に詳細に述べられているが、ここでは後述する本発明を展開する際の前提技術として簡単に説明する。

【００２３】図３５に用いられる算術符号型のエントロピー符号器５２の一例を図４０に示す。なお、算術復号型のエントロピー復号器６２の構成は、エントロピー符号器５２の構成と実質的に同一であるので、ここではその説明は省略する。

【００２４】このエントロピー符号器５２は、算術演算部５５と、状態記憶器として機能する発生確率生成手段５６とを含んで構成される。この発生確率生成手段５６内には、符号化に必要なシンボル発数確率を決定するために必要な状態パラメータテーブルが書き込まれている。上記の状態パラメータは、入力される状態信号によって特定される。そして、この状態信号によって特定された状態パラメータのテーブルに対し、発生確率生成手段５６の発生確率演算パラメータが算術演算部５５へ向けて出力される。算術演算部５５は、このようにして入力される発生確率に基づき、エントロピー符号化を行い、入力される色順位データ１２０を符号化データ２００に変換出力する。そして、符号化した色順位データ１２０の値により、状態信号に対する発生確率を再計算し、演算パラメータ更新値として、発生確率生成手段５６へ入力する。この更新結果が次データの発生確率としてテーブルに記憶されることで、エントロピー符号器５２の圧縮効率が向上することとなる。

【００２５】なお、色順位データ１２０を生成するために、先に述べたように予測器５４，６４内に色順位テーブルが配置されている。この色順位テーブルの一例として、図４２に示されるものが知られている（特開平６－２７６０４１号公報参照）。この例では、符号化対象画素Ｘに対しての色順位テーブルを決める際、２次元的な周辺画素データＲ０，Ｒ１，Ｒ２，Ｒ３を上位の色順位データとして使用し、符号化対象画素Ｘと同一ラインの１次元テーブルを下位の色順位データとして使用するも

のである。このとき、１次元テーブルから周辺画素データＲ０，Ｒ１，Ｒ２，Ｒ３のカラーシンボルを除去した後、上位と下位の色順位データをドッキングさせ符号化対象画素Ｘの色順位テーブルとしている。

【００２６】具体的にどのように色順位テーブルができあがるかを図４２に基づき説明する。符号化されるカラーシンボルが１６色の場合を考える。仮に、色順位を図４２（Ａ）に示すように、各画素の位置Ｒ０，Ｒ１，Ｒ２…Ｒ８… で固定したとき、それぞれのカラーシンボルが図４２（Ｂ）に示すように、Ｃ４，Ｃ３，Ｃ６，Ｃ５，Ｃ２，Ｃ２…のとき、できあがる最新出現表となる色順位テーブルは、図４２（Ｃ）に示すようになる。すなわち、最上位はＲ０のＣ４，２番目はＲ１のＣ３，３番目はＲ２のＣ６，４番目はＲ３のＣ５，５番目はＲ４のＣ２，６番目はＲ５にあるＣ２となるが、Ｃ２は既に発生しており、さらにＲ６のＣ４も既に発生しているので、第６番目はＲ７のＣ０となる。このようにして既に上位にある色すなわち、Ｒ０～Ｒ３に出現するカラーシンボルを除いた色順位データがＲ０～Ｒ３のデータに加わり、１６色のカラーシンボルの第１番目から１６番目までが決められる。なおこのとき、上位４つの周辺画素を学習により可変とすることもできる。

【００２７】一方、動画像の符号化や復号化においては、動き補償と呼ばれるものが知られている。この動き補償は、動画が静止画の連続であり、その１枚１枚を見ると、前のフレームや後のフレームと相関があることに着目したものである。具体的には、着目フレームを１６×１６ピクセルの複数のブロックに分割し、当該着目フレーム中の所定ブロックが参照フレームの１ピクセル左にずれた同様のブロックと似ている場合、着目フレーム中のそのブロックの画像データを符号化または復号化せず、参照フレーム中の１ピクセル左にずれたブロックに似ているというデータを符号化または復号化するものである。

【００２８】すなわち、この動き補償では、着目フレームの全データを符号化したり復号化するのではなく、変化のあった部分だけのデータを送ると共に、参照フレーム中のブロックと同一となるブロックについては、さらに動きベクトルをデータ化することによってデータ量のさらなる減少が可能となっている。

【００２９】動き補償における動きベクトルを算出する場合、Ｒ，Ｇ，Ｂからなる自然画のときには、着目フレームと参照フレームの対応するブロックのＲ，Ｇ，Ｂの差分である△Ｒ，△Ｇ，△Ｂを算出する。そして、その平均２乗誤差（$=\triangle R^2+\triangle G^2+\triangle B^2$）を得て、この値が一定値以下のときに同一ブロックと判断し、動きベクトルを生成している。また、その他に、着目フレームと参照フレームの対応するブロックの対応する位置の画素の差分の絶対値和を計算し、その最小値を求め、動きベクトルを得る方法も知られている。

【００３０】

【発明が解決しようとする課題】マルチカラー画像の場合、図４１に示されるようにインデックスの番号が近いものでもその色が大きく異なったり、インデックスの番号が遠いものでもその色が近似しているという状態が生ずる。このため、マルチカラー画像に動き補償を適用しようとしても、従来の平均２乗誤差方式や絶対値和方式では、適切にその同一性を判断できない。

【００３１】特に、色順位テーブルを利用して符号化、復号化を行うものの場合、インデックスの番号が常に変更され続けるため、動き補償を行わせる際、上述の平均２乗誤差方式等の手法は全く採用できないこととなる。

【００３２】さらに、マルチカラー画像の場合、隣接するカラーシンボルが大きく異なる場合が多く、かつその変化が目立ち易い。このため、動き補償を行うに当たり、ブロックの同一性の判断を厳しくする必要がある。ところが、この判断を厳しくすると大きなブロック間では、同一と判断されるものが極端に少なくなり、動き補償によるデータ圧縮の効果が生じなくなってしまう。一方、小さなブロックに分割して、動きベクトルを各ブロックで得るようにすると、動きベクトルのデータ量が増え、効果的な圧縮ができなくなってしまう。

【００３３】本発明は、以上のような問題に対処してなされたものであり、マルチカラー画像からなる動画像の符号化や復号化を行う際にも動き補償を適用できるようにしたマルチカラー画像からなる動画像の符号化装置およびその方法ならびにマルチカラー画像からなる動画像の復号化装置およびその方法を提供することを目的とする。また、本発明は、動き補償の際のブロックの同一性の判断を厳しくしたいが、動きベクトルの数を増やしたくないという相反する目的を同時に達成できるマルチカラー画像からなる動画像の符号化装置等を提供することを目的とする。

【００３４】

【課題を解決するための手段】かかる目的を達成するため、請求項１記載の発明では、インデックスコードが付与されたカラー画素データからなる静止画のフレームを連続させて動画像を形成する際、符号化対象の着目フレームをそれぞれＮ×Ｎピクセル（Ｎは４以上の整数）からなる複数のブロックに分割し、その着目フレーム中のブロックが参照フレーム中のブロックと同一または類似しているとき、着目フレーム中のブロック位置と参照フレーム中のブロック位置との相対変位を動きベクトルとして動き補償を行うマルチカラー画像からなる動画像の符号化装置において、動きベクトルを用いるか否かの判断手段を設け、この判断手段によって両ブロックの対応する位置の画素が異なる場合の数を算出し、その不一致数が所定数以下のときには同一または類似のブロックと判断し、動きベクトルを作成している。

【００３５】このため、マルチカラー画像からなる動画

像においても、自然画と同様に動き補償が使用できることとなり、データ圧縮が可能となる。

【００３６】また、請求項２記載の発明では、請求項１記載のマルチカラー画像からなる動画像の符号化装置において、ブロックは、ＭＮ×ＭＮピクセル（Ｍは２以上の整数）からなるより大きなマクロブロックを小ブロック化手段によってＭ×Ｍに分割され、判断手段は、各分割されたブロック毎に前記動きベクトルを用いるか否かの判断を行うために、参照フレーム内の対応するブロックの対応する位置にある画素と当該ブロックの画素とが異なる場合の数を算出し、その不一致数が所定数以下のときに同一または類似のブロックと判断し、マクロブロックの動きベクトルを利用して動き補償を行うようにしている。

【００３７】このように、より大きなブロック間で動きベクトルを作成するか否かを判断し、さらに小さいブロック間でも動き補償を行うか否か判断しているため、動きベクトルの作成数を増やすことなく動き補償の要否を厳格に判断することが可能となる。すなわち、マルチカラー画像の場合でも、データ圧縮を効果的に行えるようになる。

【００３８】さらに、請求項３記載の発明では、請求項１または２記載のマルチカラー画像からなる動画像の符号化装置において、ブロックを８×８ピクセルからなるブロックとし、所定数を０～５の整数値のいずれか１つとしている。このため、一般的な動き補償の際の１６×１６ピクセルのマクロブロックに比べ、効果的な動き検出が可能となる。しかも、判定基準が厳しいので、マルチカラー画像であっても画像の乱れは生じにくくなる。

【００３９】加えて、請求項４記載の発明では、請求項１、２または３記載のマルチカラー画像からなる動画像の符号化装置において、動きベクトルを用いないと判断したブロックを、着目フレームの符号化時の走査方向と直角となる方向に拾っていくと共に、着目画素の囲りの参照画素を利用して当該各ブロック内の各画素を符号化している。このように、動きベクトルを用いないブロックを、着目フレームの符号化時の走査方向と直角となる方向に拾っていき符号化するので、メモリが少なくて済むと共に、メモリへのアクセスが少なくなり符号化速度が向上する。

【００４０】また、請求項５記載の発明では、インデックスコードが付与されたカラー画素データからなる静止画のフレームを連続させて動画像を形成する際、符号化対象の着目フレームをそれぞれＮ×Ｎピクセル（Ｎは４以上の整数）からなる複数のブロックに分割し、その着目フレーム中のブロックが参照する参照フレーム中のブロックと同一または類似しているとき、着目フレーム中のブロック位置と参照フレーム中のブロック位置との相対変位を動きベクトルとして動き補償を行うマルチカラー画像からなる動画像の符号化方法において、動きベクトルを用いるか否かの判断を行う判断工程を設け、この判断工程によって両ブロックの対応する位置の画素が異なる場合の数を算出し、その不一致数が所定数以下のときには同一または類似のブロックと判断している。このため、マルチカラー画像からなる動画像においても、自然画と同様に動き補償が使用できることとなり、データ圧縮が可能となる。

【００４１】さらに、請求項６記載の発明は、請求項５記載のマルチカラー画像からなる動画像の符号化方法において、ブロックは、ＭＮ×ＭＮピクセル（Ｍは２以上の整数）からなるより大きなマクロブロックを小ブロック化工程によってＭ×Ｍの数に分割され、判断工程は、各分割されたブロック毎に動きベクトルを用いるか否かの判断を行うために、参照フレーム内の対応するブロックの対応する位置にある画素と当該ブロックの画素とが異なる場合の数を算出し、その不一致数が所定数以下のときに同一または類似のブロックと判断し、マクロブロックの動きベクトルを利用して動き補償を行うようにしている。

【００４２】また、請求項７記載の発明は、請求項５記載のマルチカラー画像からなる動画像の符号化方法において、ブロックは、ＭＮ×ＭＮピクセル（Ｍは２以上の整数）からなるより大きなマクロブロックを小ブロック化工程によってＭ×Ｍの数に分割され、判断行程は、各分割されたブロック毎に動きベクトルを用いるか否かの判断を行うために、参照フレーム内の対応するブロックの対応する位置にある画素と当該ブロックの画素とが異なる場合の数を算出し、その不一致数が所定数以下のときにＮ×Ｎピクセルの　ブロックについて同一または類似のブロックと判断し、マクロブロックの動きベクトルを利用して前記動き補償を行うようにしている。

【００４３】このように、請求項６または７記載の発明では、より大きなブロック間で動きベクトルを作成するか否かを判断し、さらに小さいブロック間でも動き補償を行うか否か判断しているため、動きベクトルの作成数を増やすことなく動き補償の要否を厳格に判断することが可能となる。すなわち、マルチカラー画像の場合でも、データ圧縮を効果的に行えるようになる。

【００４４】また、請求項８記載の発明は、請求項５、６または７記載のマルチカラー画像からなる動画像の符号化方法において、ブロックを８×８ピクセルからなるブロックとし、所定数を０～５の整数値のいずれか１つとしている。このため、一般的な動き補償の際の１６×１６ピクセルのマクロブロックに比べ、効果的な動き検出が可能となる。しかも、判定基準が厳しいので、マルチカラー画像であっても画像の乱れは生じにくくなる。

【００４５】加えて、請求項９記載の発明は、請求項５、６、７または８記載のマルチカラー画像からなる動画像の符号化方法において、動きベクトルを用いないと判断したブロックを、着目フレームの符号化時の走査方

向と直角となる方向に拾っていくと共に、着目画素の囲りの参照画素を利用して当該各ブロック内の各画素を符号化している。このように、動きベクトルを用いないブロックを、着目フレームの符号化時の走査方向と直角となる方向に拾っていき符号化するので、メモリが少なくて済むと共に、メモリへのアクセスが少なくなり符号化速度が向上する。

【００４６】また、請求項１０記載の発明では、入力される対象符号化データをインデックスコードが付与されたカラー画素データに復号するエントロピー復号化手段と、該カラー画素データからなる静止画を連続させて動画像を形成する動画像形成手段とを備え、動画像形成手段は、復号化対象の着目フレームをそれぞれＮ×Ｎピクセル（Ｎは４以上の整数）からなる複数のブロックに分割し、その着目フレーム中のブロックに関する復号化データ中にそのブロックの相対変位の方向を示す動きベクトルが存在するとき、参照フレーム中の同一または類似のブロックを利用してそのブロックを復号する動き補償を行うマルチカラー画像からなる動画像の復号化装置において、動きベクトルは、着目フレーム中のブロックと参照フレーム中の対応するブロックの対応する位置の画素が異なる場合のその不一致数が所定数以下のときに発生させられたものとしている。

【００４７】このため、マルチカラー画像からなる動画像の復号化においても、自然画と同様に動き補償が使用でき、復号化装置のメモリ容量を小さくできると共にデータ復号の速度を向上させることができる。

【００４８】さらに、請求項１１記載の発明は、請求項１０記載のマルチカラー画像からなる動画像の復号化装置において、ブロックを、ＭＮ×ＭＮピクセル（Ｍは２以上の整数）からなるより大きなマクロブロックから小ブロック化手段によってＭ×Ｍの数に分割し、マクロブロックの動きベクトルを利用して小さな各ブロックを復号するか否か判断する判断手段を有している。このため、より大きなマクロブロック間の動きベクトルを利用して小さなブロックの動き補償を行うことが可能になるので、動きベクトルの数を増やすことなく、小さなブロックについても動き補償が可能となる。このため、マルチカラー画像の復号化の場合でも、データのメモリ容量を小さくできると共にデータ復号を効率的に行えるものとなる。

【００４９】また、請求項１２記載の発明では、請求項１０または１１記載のマルチカラー画像からなる動画像の復号化装置において、ブロックを、８×８ピクセルからなるブロックとし、所定数を０～５の整数値のいずれか１つとしている。このため、一般的な動き補償の際の１６×１６ピクセルのマクロブロックに比べ、効果的な動き復号化が可能となる。しかも、判定基準が厳しいので、マルチカラー画像であっても画像の乱れは生じにくくなる。

【００５０】加えて、請求項１３記載の発明では、請求項１０、１１または１２記載のマルチカラー画像からなる動画像の復号化装置において、動きベクトルを有しないブロックを、着目フレームの符号化時の走査方向と直角となる方向につなぎ合わせて復号すると共にその復号の際、着目画素の囲りの参照画素を利用して当該動きベクトルを有さないブロック内の各画素を復号化している。このように、動きベクトルを用いないブロックを、着目フレームの符号化時の走査方向と直角となる方向につなぎ合わせていき復号化するので、メモリが少なくて済むと共に、メモリへのアクセスが少なくなり復号化速度が向上する。

【００５１】また、請求項１４記載の発明では、入力される対象符号化データをインデックスコードが付与されたカラー画素データに復号するエントロピー復号化工程と、該カラー画素データからなる静止画を連続させて動画像を形成する動画像形成工程とを備え、動画像形成工程は、復号化対象の着目フレームをそれぞれＮ×Ｎピクセル（Ｎは４以上の整数）からなる複数のブロックに分割し、その着目フレーム中のブロックに関する復号化データ中にそのブロックの相対変位の方向を示す動きベクトルが存在するとき、参照する参照フレーム中の同一または類似のブロックを利用してそのブロックを復号する動き補償を行うマルチカラー画像からなる動画像の復号化方法において、動きベクトルは、着目フレーム中のブロックと参照フレーム中の対応するブロックの対応する位置の画素が異なる場合のその不一致数が所定数以下のときに発生させられたものとしている。

【００５２】このため、マルチカラー画像からなる動画像の復号化においても、自然画と同様に動き補償が使用でき、この方法を採用すると、復号化装置のメモリ容量を小さくできると共にデータ復号の速度を向上させることができる。

【００５３】さらに、請求項１５記載の発明は、請求項１３記載のマルチカラー画像からなる動画像の復号化方法において、ブロックを、ＭＮ×ＭＮピクセル（Ｍは２以上の整数）からなるより大きなマクロブロックから小ブロック化工程によってＭ×Ｍの数に分割し、マクロブロックの動きベクトルを利用して小さな各ブロックを復号するか否か判断する判断工程を有している。このため、マルチカラー画像の復号化の場合でも、データのメモリ容量を小さくできると共にデータ復号を効率的に行えるものとなる。

【００５４】加えて、請求項１６記載の発明は、請求項１４または１５記載のマルチカラー画像からなる動画像の復号化方法において、ブロックを、８×８ピクセルからなるブロックとし、所定数を０～５の整数値のいずれか１つとしている。このため、一般的な動き補償の際の１６×１６ピクセルのマクロブロックに比べ、効果的な動き復号化が可能となる。しかも、判定基準が厳しいの

で、マルチカラー画像であっても画像の乱れは生じにくくなる。

【００５５】また、請求項１７記載の発明は、請求項１４、１５または１６記載のマルチカラー画像からなる動画像の復号化方法において、動きベクトルを有しないブロックを、着目フレームの符号化時の走査方向と直角となる方向につなぎ合わせて復号すると共にその復号の際、着目画素の囲りの参照画素を利用して当該動きベクトルを有さないブロック内の各画素を復号化している。このように、動き補償を行わないブロックを、着目フレームの符号化時の走査方向と直角となる方向につなぎ合わせていき復号するので、メモリが少なくて済むと共に、メモリへのアクセスが少なくなり復号速度が向上する。

【００５６】

【発明の実施の形態】本発明の実施の形態を図１から図３４に基づき説明する。なお、従来技術中のデータと対応する各データには、同一符号を付して説明する。

【００５７】図１に、本発明に係るマルチカラー画像からなる動画像の符号化システム１の好適な実施の形態を示す。また、図３に、図１の符号化システム１に対応するマルチカラー画像からなる動画像の復号化システム３の好適な実施の形態を示す。

【００５８】この符号化システム１は、動き補償を行うための動きベクトルＶを生成する動き予測部２と、入力されるカラー画素データ１００Ａから動き予測部２で生成されるデータを減算する減算器１ａと、後述するインデックス変換部７に入力するデータと動き予測部２からのデータとを加算する加算器１ｂと、静止画としてのフレームをメモリする２つのフレームメモリ１ｃ，１ｄと、各スイッチ５ａ，５ｂ，５ｃ，５ｄ，５ｅ，５ｆを制御する制御回路５と、インデックス変換部７と、周辺画素生成手段および参照画素生成手段となるラインバッファ１０と、縮退手段となるマルコフモデル生成部１１と、エントロピー符号化手段となる２つのエントロピー符号器１２，１２ａと、合成部２０とを含み、入力されるカラー画素データ１００Ａのデータストリームを符号化データ２００のデータストリームに変換して出力するように構成されている。

【００５９】減算器１ａおよび加算器１ｂによって、フレーム間の予測符号化手段を構成する。これは着目フレームに対して隣接フレームの同一位置の画素データの減算することにより、エントロピー符号器１２に入力するデータの値を０近傍の値に収め符号化効率を上昇させる手段である。フレームがＲＧＢ（Ｒｅｄ，Ｇｒｅｅｎ，Ｂｌｕｅ）それぞれの色で表現されている場合には、減算器１ａおよび加算器１ｂによって演算する値は、それぞれ色ごとに演算する。この場合は画素データ値の近傍値が類似した色となっているため、差分値は０の近傍に集中する。一方、インデックスパレットによって構成されたマルチカラー画像では、画素データ値の近傍が必ずしも類似色ではない。よってフレーム間の予測符号化は、着目フレームと隣接フレームの値が同値、すなわち同色の場合のみ０になる。このため、本発明において、前述のフレーム間予測符号化手段は必須ではない。

【００６０】次に、この動き補償の一般的な動作について説明する。カラー画素データ１００Ａは、マルチカラー画素データとして入力する。減算器１ａでは、カラー画素データ１００Ａから入力する現在の画素データと、フレームメモリ１ｃ，１ｄから転送する過去または未来の画素データとの差分を行う。これは、フレーム間予測符号化のフレーム間差分の減算器に当たる。なお、マルチカラー画像では差分によってかえって、情報量が増大する場合がある。この場合には、減算器を使用しない。減算を行わない方法としては、図１に示すように、カラー画素データ１００Ａを減算器１ａを迂回する経路に接続する方法の他、減算器を設けない方法、またはフレームメモリ１ｃ，１ｄからのデータに０情報を送る方法などがある。

【００６１】さらに、スイッチ５ａを通過したこの情報を、加算器１ｂに入力するデータと、エントロピー符号器１２に入力するデータとに分岐させる。エントロピー符号器１２に入力するデータについては後述する。加算器１ｂは、フレーム間予測符号化のフレーム間差分の加算器に当たる。加算器１ｂでは、減算器１ａで減算した過去または未来のデータと同じデータで加算を行う。よって、この加算器１ｂを通過した信号は、減算前のカラー画素データ１００Ａと同一となる。減算器１ａと加算器１ｂは、対にして、フレーム間予測符号化のフレーム差分を行うので、減算器１ａを使用しない場合には、加算器１ｂは必要なく、スイッチ５ｆによって使用しない側に切り換えられる。加算を行わないためには、図１に示すように、加算器１ｂを迂回する経路に信号を切り替える方法の他、フレームメモリ１ｃ，１ｄからのデータ線に０を入力する方法が採用される。

【００６２】この元の信号と同一の信号を、動き予測部２に入力する。元の信号が、ラスタスキャンにしたがって入力されている場合には、動き予測部２にてブロック単位の画素比較を行うために、画像バッファ（図示省略）に画素データを入力する。この場合には、画像バッファは少なくとも、ブロックの縦の画素数×画像フレームの横画素数分の画像データを入力できる容量になる。この画像バッファの値と、フレームメモリ１ｃ，１ｄに格納された過去または未来の画像データと排他的論理和ＸＯＲにて、該当画素が互いに一致しているかどうかを検証する。この検証は、画像バッファ中のブロックと同じ座標の、フレームメモリ１ｃ，１ｄ中のブロックを取り出しＸＯＲ比較する他に、その同じ座標の周囲のブロックについても比較する。

【００６３】画素データは、マルチカラー画像では、１

画素あたり１バイト程度のデータから成り立つ。排他的論理和ＸＯＲは、フレーム画像データの画素と、画像バッファの画素が同一であるかどうかの確認であるので、１画素を表す全てのビットをＸＯＲするのではなく、どれか１ビットでも異なっているものがあれば、確認は終了する。

【００６４】画素の一致、不一致は、動き予測部２内のカウンター（図示省略）によって、カウントする。ここでは不一致検出ごとにカウンターが増大するとする。このカウンターは、フレームメモリ１ｃ，１ｄの１ブロック分のカウントを終了するごとにリセットされる。リセット直前のカウント値は、カウンターに内在している、カウント値の配列格納部（図示省略）に、現在の比較しているブロックの位置と、フレームメモリ１ｃ，１ｄのブロック位置とのずれごとに格納する。

【００６５】現在の画像の１ブロック分の比較の終了時には、配列格納部中でカウント値のもっとも小さい値を検索する。その値が設定した閾値よりも大きい場合には、現在のブロックが過去または未来の該当位置のブロックと、同一または類似していないと判別する。判別の結果によって行う動作は、順方向予測すなわち過去から現在方向の予測と、逆方向予測すなわち未来から現在方向の予測と、双方向予測すなわち過去または未来から現在への予測とでは動作が異なる。

【００６６】順方向予測では、同一または類似であると判別した時、過去フレームの最も不一致点が少ない領域を、現在のフレームへコピーする。逆に同一または類似ではない時には、画像バッファから現在のフレームにコピーする。このとき、動き予測部２以外の部分では、画像バッファの比較したブロックに相当するデータをエントロピー符号器１２によって符号化する。

【００６７】逆方向予測では、同一または類似であると判別した時、未来フレームの最も不一致点が少ない領域を、現在のフレームへコピーする。逆に同一または類似ではない時には、画像バッファから現在のフレームにコピーする。このとき、動き予測部２以外の部分では、画像バッファの比較したブロックに相当するデータをエントロピー符号器１２によって符号化する。

【００６８】双方向予測では、同一または類似でないと判別した時、動き予測部２以外では、画像バッファの比較したブロックに相当するデータをエントロピー符号器１２によって符号化する。同一または類似であると判別した時は、順方向予測または逆方向予測と同一となる。

【００６９】順、逆および双方向予測ともに、同一または類似であると判別した時の、最も不一致点が少ない領域と、現在判断している領域との相対位置のずれを、動きベクトルＶとする。この動きベクトルＶは、動きベクトルＶ用に決める符号化手順により圧縮や多重化を行う。

【００７０】ここで、ブロックをＮ画素×Ｎ画素単位とすると、このブロックの上位のM^2（Ｍは２以上の整数）個のＮ×Ｎ単位のブロックによってＭＮ×ＭＮのマクロブロックＭＢを構成し、マクロブロックＭＢごとに動きベクトルＶを使用するか否か判断してもよい。その場合には、ブロックごとの判断の閾値の他に、マクロブロックＭＢに対する閾値を定めて、その閾値に対しての判断を行う方法の他に、ブロックとマクロブロックＭＢの双方の閾値を使う方法がある。

【００７１】次に、この動き予測部２の具体的な動きを図を参照しながら説明する。従来と同様に、全く予測を用いないフレームをＩピクチャー、過去の画像を用いて予測を行うフレームをＰピクチャー、過去と未来の画像を利用して予測を行うフレームをＢピクチャーとすると、例えば図５（Ａ）に示すように、入力順は、ＰＢＢＰＢＢＩＢＢＰＢＢＰＢＢＩとなる。このように、各フレームに役割を与えると、符号化順序は図５（Ｂ）に示すとおりとなる。すなわち、例えば、Ｎｏ．８のフレームは、Ｎｏ．７のＩピクチャーとＮｏ．１０のＰピクチャーが符号化された後に、それらのデータを利用して符号化される。この符号化の際、動き補償が適宜使用される。

【００７２】動き補償は、ＰピクチャーとＢピクチャーを符号化するときに使用される。具体的には、符号化しようとするフレーム内を、図６に示すように、１６×１６ピクセルのマクロブロックＭＢに分割し、そのマクロブロックＭＢと同一と判定されるマクロブロックを参照フレームから捜し出す。なお、参照フレームは、Ｐピクチャーを符号化するときはＩピクチャーで、Ｂピクチャーを符号化するときはＩピクチャーとピクチャーとなる。

【００７３】参照フレームから同一ブロックを捜し出す方法は、従来と同様に、参照フレームの対応するマクロブロックＭＢの位置およびその位置からわずかにずれた位置のマクロブロックＭＢの各画素を対比することにより行う。この実施の形態では、同一か否かの判断を両マクロブロック内の対応する位置の画素が異なる場合のその個数で行っている。すなわち、不一致数が所定値以下となるときに、動き補償部２の判断手段２ａが両マクロブロックＭＢを同一と判断する。この実施の形態では、このマクロブロックＭＢの判定のための所定値を２０個としている。

【００７４】同一と判断すると、動き予測部２は、両マクロブロックＭＢの移動方向をベクトル化した動きベクトルＶを生成し、その値をハフマンテーブルを利用した一方のエントロピー符号器１２ａへ送出する。動き予測部２は、マクロブロックＭＢを４つに分割して小さな４つのブロックＳＢを生成する小ブロック化手段２ｂを有している。そして、このちいさなブロックＳＢについても同一か否かを判断手段２ａにて判定している。この実施の形態では、そのブロックＳＢと参照フレーム中のブ

ロックＳＢの画素の不一致数が３個以下のとき、両ブロックＳＢは同一と判断している。同一か否かの判定基準となる所定数としては、０～５個の値とするのが、マルチカラー画像の乱れが生じずかつ効果的な圧縮を行うためには好ましい。なお、ブロックＳＢはここではマクロブロックＭＢを４つに分割しているが、そのブロック数は４つに限るものではなく、また最も小さいブロックとしては、１画素も許される。

【００７５】なお、この小さなブロックＳＢについては、動きベクトルＶ用のデータ部分を設けず、先のマクロブロックＭＢの動きベクトルＶをそのまま利用する。このため、動きベクトルＶ用のデータ部分は小さくなり、データ増大を招くことがない。一方、同一か否かの判定は、小さなブロックＳＢで厳しく行うことができるので、マルチカラー画像の乱れも生じない。

【００７６】例えば、図６に示すマクロブロックＭＢと、参照フレームのマクロブロックＭＢとの画素の不一致数が仮に１８個とすると、両マクロブロックＭＢは同一と判定され、動きベクトルＶが生成される。一方、小さなブロックＳＢ（１）の不一致数が３個、ＳＢ（２）が２個、ＳＢ（３）が２個とすると、小さなブロックＳＢ（１），ＳＢ（２），ＳＢ（３）は同一のものがあると判定され、マクロブロックＭＢによって生成された動きベクトルＶを利用してデータ圧縮される。しかし、小さなブロックＳＢ（４）の不一致数は１１個となることとなり、この小さなブロックＳＢ（４）は、動き補償を使用することなく、その画素が符号化される。

【００７７】この実施の形態では、マクロブロックＭＢの同一判定を甘くし、小さなブロックＳＢの同一判定を厳しくしている。このため、動きベクトルを利用できるブロックが増える一方、画像の乱れも生じないものとなる。なお、画像の性質によっては、マクロブロックＭＢの同一判定を厳しくし、小さなブロックＳＢの同一判定を甘くしたり、両者を同一レベルのものとしても良い。

【００７８】インデックス変換部７は、図２に示すように、参照画素生成手段となるラインバッファ１０と、縮退手段となるマルコフモデル生成部１１と、エントロピー符号化手段となるエントロピー符号器１２とに接続され、状態信号ＳＴを生成する状態生成部１３と、プリスキャン時に２種類の変換テーブルを生成する変換テーブル生成部１４と、参照順位テーブル１５と、符号化時に画素インデックス（カラーシンボル）３を入力し、インデックス変換する画素インデックス変換テーブル１６と、カラーパレット変換部１７と、変換された変換カラーパレット１８と、変換された画素インデックスを入力し、所定の順位を出力する判別部１９とを含むように構成されている。

【００７９】なお、この実施の形態において、符号化の対象とするカラー画素は、マルチカラーの画素インデックス１００Ａであり、１画素当たり８ビットのインデックスコードのデータで構成され、２５６色分のカラーシンボルを表示できる。

【００８０】ラインバッファ１０は、プリスキャン時には、参照画素を状態生成部１３に入力し、符号化時には参照画素をマルコフモデル生成部１１に入力している。マルコフモデル生成部１１は、図７（Ａ）に示すように、符号化対象画素ｂｕｆ〔ｉ〕に対し参照画素として、周辺の４画素Ｒ〔０〕～Ｒ〔３〕を取り入れ、マルコフ状態信号ＣＸを発生させている。

【００８１】エントロピー符号器１２は、後述するように、予測ランレングス符号化によってモデル化されて画像信号を符号化コード化するものとなっている。このエントロピー符号器１２は、図４０に示す算術型のエントロピー符号器や、ハフマン符号を使用した符号器を採用するようにしても良い。なお、エントロピー符号器１２の構成、動作、機能の詳細については後述する。

【００８２】状態生成部１３は、符号化プロセスに先立つプリスキャン時に動作するもので、図７（Ａ）に示すように、符号化対象画素をｂｕｆ〔ｉ〕としたとき、その周辺の参照画素Ｒ〔０〕～Ｒ〔０３〕の状態を図８に示すように区分けするものとなっている。例えば、Ｒ〔０〕～Ｒ〔３〕がすべて同一の色の時は、状態信号ＳＴは「０」となり、Ｒ〔０〕～Ｒ〔３〕がすべて互いに異なる色のときは状態信号ＳＴは「３３」となる。なお、符号化対象画素ｂｕｆ〔ｉ〕は、先頭ラスタにあるときは、図７（Ｂ）に示すように、現れてこない部分である参照画素Ｒ〔１〕～Ｒ〔３〕にはすべて「０」、すなわち後述するように最も頻度の高い画素インデックスを設定する。また、符号化対象画素ｂｕｆ〔ｉ〕が、画像の先頭であるとき、Ｒ〔０〕～Ｒ〔３〕のすべてに「０」を設定する。

【００８３】変換テーブル生成部１４も、プリスキャン時にのみ動作する。そして、符号化対象画素ｂｕｆ〔ｉ〕が４つの参照画素（以下Ｒ〔４〕という）と一致した場合、状態信号毎にその一致した色の位置の度数を算出すると共に、参照画素Ｒ〔４〕の中に符号化対象画素ｂｕｆ〔ｉ〕が無いときには、その一致しなかった画素インデックスの出現度数をカウントする。

【００８４】例えば、状態信号ＳＴが「３」のとき、図９（Ａ）に示すように、符号化対象画素ｂｕｆ〔ｉ〕が参照画素Ｒ〔０〕と一致した数をＮＡとし、参照画素Ｒ〔２〕と一致した数をＮＢとすると、ＮＡ＞ＮＢなら作成する参照順位テーブル１５の参照画素位置は、図９（Ｂ）に示すように、０位にＲ〔０〕がきて、１位にＲ〔２〕がくる。一方、ＮＡ＜ＮＢのときは、図９（Ｃ）に示すように、０位にＲ〔２〕、１位にＲ〔０〕がくる。このようにして、図１０に示すような参照順位テーブル１５が作成される。なお、参照順位テーブル１５中の数字、例えば、状態信号ＳＴが「３」のときの０位の「２」、１位の「０」は、先に示した例で言えば、０位

のR〔2〕の「2」を、1位のR〔0〕の「0」を示している。また、状態信号ＳＴが「0」のときは0位のみであり、参照順位テーブル１５から除かれるため、そのテーブル１５は縦が１４個、横が4個のテーブルとなっている。

【００８５】また、変換テーブル生成部１４は、参照画素R〔4〕の中に符号化対象画素ｂｕｆ〔ｉ〕の色が無いときは、図１１に示すように、参照画素R〔4〕中になかった度数を画素インデックス毎にカウントする。そして、図１２に示すような画素インデックス変換テーブル１６を生成する。なお、図１２に示すものは、N2≧N0 ≧…≧Nn の場合となっている。

【００８６】カラーパレット変換部１７は、カラーパレット（例えば、図１３の左側に示すようなカラーパレット）9を、画素インデックス変換テーブル１６を利用して図１３（B）に示すような変換カラーパレット１８を生成している。例えば、画素インデックス（カラーシンボル）C2 がＲＧＢの三原色の組み合わせとしてＲＧＢ2 なるものとされているとき、参照画素R〔4〕中に無かった度数の順位が1番目すなわち順位として0順位のとき、変換カラーパレット１８では最上位すなわちカラーシンボルC0 の位置に設定されることとなる。

【００８７】判別部１９は、符号化時に動作するもので、画素インデックス変換テーブル１６によって変換された画素インデックスと、状態信号ＳＴに基づく参照画素が入力される。そして、その画素インデックスと参照画素が一致したときは、参照順位テーブル１５に基づいてその順位を出力する。一致しなかった場合は、その画素インデックスの順位に、マルコフ状態信号ＣＸに1を加えた数を加算して出力する。例えば、変換された画素インデックスがCn のときで状態信号ＳＴが「3」の場合は、その画素インデックスCn と同一の色が参照画素R〔0〕にあったときは、順位として「1」を出力する。一方、画素インデックスCn が参照画素R〔4〕中に無いときは、ＣＸが「1」なので、「n＋1＋1」＝「n＋2」を出力する。

【００８８】エントロピー符号器１２は、判別部１９の出力をマルコフ状態信号ＣＸでマルコフモデル化し、可変長符号化する。合成部２０では、変換カラーパレット１８のデータと、参照順位テーブル１５のデータと、エントロピー符号器１２からの出力とを合成して符号化データ２００として出力する。この実施の形態では、符号化データ２００は、画像のサイズ等の情報に続き、変換カラーパレット１８のデータ、参照順位テーブル１５のデータ、最後にエントロピー符号器１２によって符号化された符号化コードという順に出力されるが、基本的には符号化データ２００中に入っていれば良く、他の順序としても良い。

【００８９】次に、以上の構成を有する符号化システム1の動作について、図１４および図１５に示すフローチャートに基づいて説明する。なお、動き予測部2の動作は、既に説明したので、ここではインデックス変換部7およびその周辺の部材の動作について説明する。

【００９０】符号化システム1の動作は、プリスキャンプロセスと符号化プロセスとに分かれる。符号化の前処理に当たるプリスキャンプロセスでは、後段の符号化プロセスで適切な符号化が行えるように、画素インデックス変換テーブル１６および参照順位テーブル１５の作成を行う。このために、まず、スイッチ２１を端子２１ａに接続する。そして、ステップＳ０では変換テーブル生成部１４中の後述するカウントテーブルならびに参照順位テーブル１５および画素インデックス変換テーブル１６の初期化ならびに後述する符号化処理のための各種変数の初期化を行う。

【００９１】ステップＳ１では、入力画像のカラーパレット9のデータをカラーパレット変換部１７中の配列ｐａｌｌｅｔ〔　〕に入力する。ステップＳ２では、入力画像の画素インデックス１００Ａをラインバッファ１０中の配列ｂｕｆ〔　〕に入力する。次に、ステップＳ３では、符号化対象画素ｂｕｆ〔ｉ〕の周辺の4画素をラインバファ１０から取り出し、変換テーブル生成部１４や状態生成部１３等の配列R〔　〕に入力する。符号化対象画素ｂｕｆ〔ｉ〕と参照画素R〔0〕～R〔3〕の位置関係は図7に示すとおりである。ステップＳ４では、ラインバファ１０から抽出した参照画素に基づき、図8の基準で状態信号ＳＴを生成する。

【００９２】ステップＳ５では、参照画素R〔4〕の中に符号化対象画素ｂｕｆ〔ｉ〕と一致するものがあるかどうか調べる。その後、ステップ6で参照画素R〔ｉ〕と一致した場合、変換テーブル生成部１４中の参照順位テーブル１５用のカウントテーブルN__ｔａｂｌｅ__A〔ＳＴ＋ｊ〕に1を加える。ここで、ｊは参照画素の位置を現すため、N__ｔａｂｌｅ__A〔　〕には、参照画素の状態と一致した位置毎に、一致した回数がカウントされる。次のステップＳ７では、参照画素と一致しない場合、変換テーブル生成部１４中の画素インデックス変換テーブル１６作成用のカウントテーブルN__ｔａｂｌｅ__B［ｂｕｆ〔ｉ〕］に1を加える。つまり、カウントテーブルN__ｔａｂｌｅ__B［ｂｕｆ〔ｉ〕］には、参照画素と一致しなかった画素について、その画素インデックス3ごとにカウントされる。

【００９３】ステップＳ８で、すべての画素データについて、ステップＳ３からステップ6あるいはステップ7までの処理を繰り返す。カウントテーブルN__ｔａｂｌｅ__B〔　〕には、画素インデックス１００Ａ毎に出現度数がカウントされている。ステップＳ９では、これを基に出現度数の多いものから昇順のコードに変換する画素インデックス変換テーブル１６を作成する。このテーブル１６により、出現度数の多い画素インデックス１００Ａほど、小さいインデックス値に変換され、その特性

に合わせたエントロピー符号化を用意すれば、いかなる画像であっても最も効率の良いエントロピー圧縮を行うことができる。

【００９４】ステップＳ１０では、その参照順位を記述した参照順位テーブル１５を作成する。すなわち、カウントテーブルN__ｔａｂｌｅ__Ａ〔　〕には、参照画素の状態信号ＳＴと、符号化対象画素ｂｕｆ〔ｉ〕に一致した参照画素位置毎にその度数がカウントされているので、度数の多い順に昇順を付ければ、最も効率の良い圧縮が期待できる。

【００９５】プリスキャンプロセスによって、参照順位テーブル１５と画素インデックス変換テーブル１６を作成後、図１５に示す符号化プロセスを実行する。この実行に当たり、まずスイッチ２を端子２１ｂ側に切り換える。

【００９６】プリスキャンプロセスにて作成された参照順位テーブル１５は、復号処理に必要となる。このため、圧縮ストリームのヘッダ情報の一部として符号化データ２００のストリームに付加するために、ステップＳ１１で、合成部２０に出力される。ステップＳ１２では、プリスキャンプロセスで作成した画素インデックス変換テーブル１６を用いて、画素インデックス１００Ａの変換を行う。この変換により参照画素と一致しなかった画素インデックス１００Ａが、その出現度数の大きい順に昇順のコードとなる。

【００９７】画素インデックス１００Ａの変換に伴って、カラーパレットも復号側で正しい色が再現されるように変換する必要がある。このため、ステップＳ１３で、画素インデックス変換テーブル１６を利用して、カラーパレット変換部１７で変換カラーパレット１８の作成を行う。変換された変換カラーパレット１８は、合成部２０に入力し、符号化データ２００からなる圧縮ストリームのヘッダ情報の一部として付加される。ステップＳ１４では、プリスキャンプロセスのステップＳ３と同じ動作をする。すなわち、ラインバッファ１０から参照画素Ｒ〔４〕を取り出し、マルコフモデル生成部１１や状態生成部１３に入力する。

【００９８】次のステップＳ１５では、プリスキャンプロセスのステップＳ４と同じ動作をする。すなわち、状態生成部１３で、図８に示す基準に基づいて状態信号ＳＴを、参照順位テーブル１５に向けて出力する。ステップＳ１６では、マルコフ状態信号ＣＸを生成する。このマルコフ状態信号ＣＸは、参照画素の色数を現す値で、図８の表に基づいて参照画素の状態から生成する。なお、この符号化プロセスのフローでは明記しないが、このマルコフ状態信号ＣＸは、エントロピー符号化を行う際のマルコフモデル化のための符号化条件としても用いている。

【００９９】ステップＳ１７では、符号化対象画素ｂｕｆ〔ｉ〕が、参照画素Ｒ〔４〕と一致するか否か調べる。具体的には、参照順位テーブル１５中の状態信号ＳＴに対応する欄内を０からＣＸと等しい値まで変化させ、符号化対象画素ｂｕｆ〔ｉ〕と一致するか調べていく。参照画素と一致した場合は、即ステップＳ１８に進み、そうでない場合はステップＳ１９に進む。

【０１００】参照画素と一致した場合は、その時の順位をエントロピー符号化する（ステップＳ１８）。参照画素と一致しなかった場合は、符号化対象画素ｂｕｆ〔ｉ〕を直接エントロピー符号化する。ただし、（ＣＸ＋１）個の符号が参照画素と一致した場合の符号に割り当てられているので、（ＣＸ＋１）を加算して符号化する（ステップＳ１９）。ステップ２０で、すべての画素データについて、ステップＳ１４からステップＳ１８あるいはステップＳ１９までの処理を繰り返す。

【０１０１】ところで、上述したフローでは、ステップＳ１８あるいはステップＳ１９で順位の確定と共に逐次エントロピー符号化を行うようになっているが、全画素の順位と対応するマルコフ状態信号ＣＸを一旦バッファに格納し、まとめて後述する符号化を行っても良い。例えば、後述する予測ランレングスの符号化方法を採用した場合、符号化対象シンボル以降の未来のシンボルをまとめて符号化することとなる。

【０１０２】なお、ＰピクチャーやＢピクチャーの符号化において、所定のマクロブロックＭＢが動き補償を行わないと判断されたとき、それらのマクロブロックＭＢが拾われ、連続して符号化される。例えば、図１６（Ａ）に示すように、マクロブロックＭＢのＮｏ．１～Ｎｏ．７が動き補償を行わないものであるとき、符号化はＮｏ．１→Ｎｏ．４→Ｎｏ．２→Ｎｏ．６→Ｎｏ．７→Ｎｏ．５→Ｎｏ．３の順で行われる。このため、メモリ容量を小さくしても効果的な圧縮が可能となる。動き予測の結果、動きベクトルＶを利用しないで、エントロピー符号化を行い復号化システム３に伝送する場合、マクロブロックＭＢの符号化の再には、図１６（Ｂ）のようにマクロブロックＭＢ内をラスタスキャンによって画素値をラインバッファ１０に格納しつつ符号化する。例えば、図１６（Ｂ）に示す、Ｎｏ．１，４，２のマクロブロックＭＢをＮｏ．１→Ｎｏ．２の順で符号化すると、Ｎｏ．２のマクロブロックＭＢのＮｏ．１と隣接する部分の画素を効果的に圧縮するには、Ｎｏ．１のマクロブロックＭＢの右端側の少なくとも１列について記憶するためのメモリが余分に必要となると共にアクセスの手間がかかる。ところが、Ｎｏ．１→Ｎｏ．４の順で符号化すると、Ｎｏ．４のマクロブロックＭＢのＮｏ．１と隣接する部分の画素の符号化には先に示した最新出現表をそのまま利用でき、余分なメモリが必要とならずアクセス時間も速くなる。

【０１０３】なお、復号化システム３に伝送されるブロックは、１度エントロピー符号化をかけられるが、動きベクトルＶを使用することによって画像そのものを復号

化システム３に伝送しなくなるブロックは１度もエントロピー符号化がかけられないこととなる。

【０１０４】次に、符号化システム１に対応したマルチカラー画像からなる動画像の復号化システム３について説明する。

【０１０５】この復号化システム３は、動き補償部４と、後述するインデックス変換部８からのデータと動き補償部４からのデータとを加算する加算器３ａと、２つのフレームメモリ３ｂ，３ｃと、各スイッチ６ａ，６ｂ，６ｃ，６ｄを切り替える制御回路６と、インデックス変換部８と、周辺画素生成手段および参照画素生成手段となるラインバッファ３０と、縮退手段となるマルコフモデル生成部３１と、エントロピー復号化手段となる２つのエントロピー復号器３２，３２ａと、分離部３３とを含み、入力される符号化データ２００のデータストリームをカラー画素データ１００Ｂのデータストリームに変換して出力するように形成されている。

【０１０６】このとき、復号化システム３のアルゴリズムは、符号化システム１のアルゴリズムと全く逆のアルゴリズムになるように構成されている。したがってカラー画素データ１００Ａのビット構成およびデータストリームと、カラー画素データ１００Ｂのビット構成およびデータストリームとは全く同じものとなる。また、動き補償部４は、先の符号化システム１の動き予測部２と基本的に同一となっており、その内部に、動きベクトルＶの有無を判断する判断手段４ａを有している。

【０１０７】すなわち、動き補償部４の動作は、動き予測部２と逆の動作を行う。動きベクトルＶの情報を元にして、動きベクトルＶと、その動きベクトルＶを使用するか否かの情報を入手する。動きベクトルＶを使用する場合には、フレームメモリ３ｂ，３ｃ中のデータを復号し、動きベクトルＶを使用しない場合には、エントロピー復号器３２から復号したデータを、復号データとして採用する。

【０１０８】動きベクトルＶを使用する場合、順方向のときには過去のデータから、逆方向のときには未来のデータから、それぞれデータを転送する。双方向の場合には、過去または未来のいずれかを選択し転送する。一方、動きベクトルＶを使用しない場合には、エントロピー復号器３２からのデータを復号データとして出力する他に、順、逆方向の場合には、現在のフレームデータとしてフレームメモリ３ｂ，３ｃに格納する。

【０１０９】フレーム間予測符号の差分符号化に関しては、復号化システム３には加算器３ａを含むこととなる。符号化システム１の構成に準じて、加算器３ａを迂回させたりまたは、加算器３ａのもう一方の入力に０を入力することによってフレーム間予測符号の差分符号化を回避させることができる。

【０１１０】インデックス変換部８は、参照画素生成手段となるラインバッファ３０と、縮退手段となるマルコフモデル生成部３１と、エントロピー復号化手段となるエントロピー復号器３２とに接続され、状態信号ＳＴを生成する状態生成部３４と、参照順位テーブル３５と、画素インデックス（カラーシンボル）生成部３６とを含むように構成されている。

【０１１１】ラインバッファ３０は、符号化システム１のラインバッファ１０と同様に複数の画素を保存できるものとなっている。そして、その値を周辺画素としてマルコフモデル作成部３１や状態生成部３４等に入力し、参照順位テーブル３５や各信号ＳＴ、ＣＸを作成している。

【０１１２】エントロピー復号器３２は、マルコフ状態信号ＣＸを利用して、入力する符号化データ２００中の符号化コードを、エントロピー符号器１２と逆の手順で復号化演算処理する。なお、エントロピー復号器３２は、エントロピー符号器１２とは全く逆のアルゴリズムでその演算を行うように形成しなければならない。したがって、エントロピー復号器３２は、エントロピー符号器１２に算術符号器が用いられた場合には、それと同様な構成の算術復号器として形成する必要があり、また、エントロピー符号器１２にハフマン符号器が用いられた場合には、それと同じ構成のハフマン復号器として構成する必要がある。

【０１１３】分離部３３は、符号化データ２００のデータストリームより変換カラーパレット１８のデータと、参照順位テーブル３５のデータとを分離するものである。そして、その内部に符号化システム１の合成部２０と逆の機能が設定されており、エントロピー符号器１２で符号化された符号化コードをエントロピー復号器３２に出力する。

【０１１４】状態生成部３４は、符号化システム１中の状態生成部１３と同様な構成、機能を有しており、図８に示す表に示された基準に基づいて状態信号ＳＴを出力する。参照順位テーブル３５は、分離部３３で分離された参照順位テーブル３５用のデータによって構成され、符号化システム１中の参照順位テーブル１５と同様な表となっている。画素インデックス生成部３６は、符号化システム１で生成された所定順位のデータおよびマルコフ状態信号ＣＸならびに参照順位テーブル３５の各データから画素インデックス６を出力する。

【０１１５】次に、このように構成される復号化システム３の動作について、図１７に示すフローチャートに基づいて説明する。

【０１１６】まず、ステップＳ３０で、参照画素や後述する予測ランレングス（ＰＲＬＣ）復号のための各種変数の初期化を行う。次に、分離部３３によって符号化データ２００の圧縮ストリームのヘッダに付加された参照順位テーブル３５のデータを抽出し参照順位テーブル３５を生成する（ステップＳ３１）。ステップＳ３２では、圧縮ストリームとなる符号化データ２００のヘッダ

に付加された変換カラーパレット１８のデータを同様に分離部３３によって抽出し出力する。

【０１１７】次に、ステップＳ３３で、参照画素Ｒ〔4〕を抽出する。これは符号化プロセスのステップＳ１４と全く同様となっている。ステップＳ３４では、マルコフ状態信号ＣＸを生成する。これは符号化プロセスのステップＳ１６と全く同様となっている。ステップＳ３５では、ＰＲＬＣによるエントロピー復号により復号順位Ｘを得る。

【０１１８】ステップＳ３６では、画素インデックス生成部３６において、順位Ｘがマルコフ状態信号ＣＸの値以下かどうかを判定する。順位Ｘが値ＣＸ以下の場合、復号される画素インデックスが参照画素に存在することを示し、ステップＳ３７に進む。それ以外は、ステップＳ３８に進む。ステップＳ３７では、状態信号ＳＴで区分される中に存在する順位Ｘが指し示す位置にある参照画素の画素インデックス１００Ｂを出力する。

【０１１９】ステップＳ３８では、復号順位Ｘが参照画素を指し示さない場合は、復号順位Ｘそのものが復号される画素インデックス６となる。ただし、（ＣＸ＋１）個の符号が参照画素に割り当てられているため、復号順位Ｘから（ＣＸ＋１）を差し引く必要がある。ステップＳ３９で、すべての画素インデックス６の復号が完了するまで、ステップＳ３３からステップＳ３７あるいはステップＳ３８までを繰り返す。そして、すべての画素インデックス６の復号が終了すると、復号動作が終了する。

【０１２０】次に、先に説明した符号化システム１中のエントロピー符号器１２で行う符号化について説明する。この実施の形態では、いわば予測ランレングス（以下、ＰＲＬＣという）と称するものを採用している。このＰＲＬＣは、良く知られているランレングス符号化を改良したもので、そのアルゴリズムの概要について、図１８から図２０に基づいて説明すると共に、このＰＲＬＣの基礎となる基本的な符号化方法等を図２１および図２２に基づいて説明する。

【０１２１】この符号化の基本のアルゴリズムは、従来から良く知られているＱＭコーダと同様、基本的には、２値のビット列を圧縮の対象としている。まず初期値として、“０”か“１”のいずれかを優勢シンボルと定め、そのシンボルが連続すると予測する個数ｒｕｎを設定する。入力系列の出現確率が不明の場合は、ｒｕｎを１に設定するのが良い。その上で、以下に示すようなルールに従い符号化を進める。なお、個数ｒｕｎが予測ビット数に相当する。

【０１２２】図１８に示すように、ｒｕｎで示される注目系列がすべて優勢シンボルであると予測し、予測が当たったとき、符号語として“０”を出力し、この系列の符号化を完了する。はずれた場合は“１”を出力し、次の分割符号化工程を実行する。

【０１２３】予測がはずれた場合は、図１９に示すように注目系列を前半部系列と後半部系列の２つに分け、前半部がすべて優勢シンボルのときは符号語として“０”を、前半部系列に劣勢シンボルが存在し後半部がすべて優勢シンボルのときは、符号語として“１０”を、前半部にも後半部にも劣勢シンボルが存在するときは“１１”をそれぞれ出力する。そして、劣勢シンボルが存在する系列は、さらに系列を分割する（図２０参照）。劣勢シンボルが存在する系列は、可能な限り系列を分割して上述の分割符号化工程を繰り返す。

【０１２４】なお、分割は必ずしも２つの均等分割とする必要はなく、不均等な分割としたり３つ以上の分割としても良い。また、予測が当たったとき“０”ではなく、優勢シンボルを出力し、はずれた場合“１”ではなく、劣勢シンボルを出力するようにしたり、予測当たりで“１”を、予測はずれで“０”を出力するようにしても良い。

【０１２５】以上がこの符号化の前提となるデータ符号化の基本アルゴリズムであるが、さらに、入力系列の出現確率の変化に追随し、符号化効率を向上させるため、以下の処理を加えるようにしても良い。

【０１２６】すなわち、ｒｕｎで予測した系列が続けて所定回数、例えば、２回当たったとき、ｒｕｎを２倍等に増加させる。なお、予測が的中し続けた場合、さらに予測範囲を拡大していく。一方、ｒｕｎで予測した系列が２回以上劣勢シンボルを含むと共にｒｕｎを１／２倍する。予測がはずれ続けた場合、さらに１／２倍していく。そして、ｒｕｎが１で、それが劣勢シンボルのときは、以降の入力系列を反転させる。すなわち、優勢シンボルを変更させる。

【０１２７】なお、ｒｕｎで予測した系列の後半部系列に劣勢シンボルが存在するとき、ｒｕｎをいきなり１／４等に減少させるようにしても良い。これは、後半部に劣勢シンボルが存在するときは、次に続く系列に劣勢シンボルが多く含まれると判断されているためである。このため、ｒｕｎで予測した系列の前半部系列のみに劣勢シンボルが存在するときは、後半部に劣勢シンボルが存在するときより多い値、例えばｒｕｎを１／２倍するようにしても良い。

【０１２８】この実施の形態のエントロピー符号器１２内の符号化プロセスは、次に説明する図２１および図２２の符号化プロセスを改良したものであり、まずその符号化プロセスについて説明する。改良前の符号化プロセスは、図２１の符号化メインルーチンと図２２の符号化サブルーチンにより構成される。なお、図２２中の符号化サブルーチンは、サブルーチンから同じサブルーチンを呼び出すいわゆる関数の再帰読出しを行っている。

【０１２９】まず、図２１の符号化メインルーチンの各ステップについて説明する。なお、符号化の対象は２値のビット列からなる入力系列となっている。最初に、予

測の初期値ｒｕｎの設定と優勢シンボルの選択（“０”または“１”）を行う（ステップＳ５０）。次に、ローカル変数ｏｆｓに０を、ｗｉｄｔｈにｒｕｎを代入する（ステップＳ５１）。ここでｏｆｓは、符号化のために予め定義した配列Ａのポインタで、予測開始ビット位置を示す。したがって初期値は０となる。ｗｉｄｔｈはｏｆｓで示したビット位置から何ビットを予測の対象にするかを示す値で、ここでは、予測の初期値ｒｕｎが代入される。その後、予め定義した配列ＡのＡ〔ｏｆｓ〕からＡ〔ｗｉｄｔｈ－１〕までに入力ビットを書き込む（ステップＳ５２）。そして、Ａ〔ｏｆｓ〕からＡ〔ｗｉｄｔｈ－１〕のすべての要素が優勢シンボルのときステップＳ５４へ進み、ひとつでも劣勢シンボルが含まれているときは、ステップＳ５５へ進む。

【０１３０】予測が的中した場合、符号語として予測当たり信号“０”を出力し、配列Ａに取り込んだ系列の符号化を完了する（ステップＳ５４）。一方、予測はずれた場合、符号語として予測はずれ信号“１”を出力する（ステップＳ５５）。そして、ｗｉｄｔｈが１以上か否かを検出する（ステップＳ５６）。ｗｉｄｔｈが１以下ならこれ以上分割できないので、ステップＳ７の符号化サブルーチンへは移行せずステップＳ５８へ移行する。一方、ｗｉｄｔｈが１を超えていると、図２２の符号化サブルーチンを呼び出す（ステップＳ５７）。

【０１３１】ステップＳ５８では、予測ｒｕｎの再設定と必要ならば優勢シンボルの変更を行う。すなわち、このステップＳ５８においては、基本的には予測が的中すれば、ｒｕｎを大きくし、はずれれば小さくする。そしてｒｕｎを小さくしても予測が所定回数はずれ続けるようなら、優勢シンボルの変更を行う。なお、予測の的中や予測のはずれをどのように評価するかについては、さまざまな方法を採用することができる。たとえば、予測がはずれた場合、直ちにｒｕｎを小さくしたり、２回以上連続してはずれたとき、初めてｒｕｎを小さくする等の方法を採用することができる。さらに、前半部系列もしくは後半部系列のみはずれた場合と、両方はずれた場合とでｒｕｎの縮小の度合いを異ならせる方法も採用できる。また、符号済みビット系列で所定の確率テーブルを引き、次の予測ｒｕｎを設定する等の方式も採用可能である。

【０１３２】符号化メインルーチンで１次予測がはずれた場合は、ステップＳ５７で図２２に示す符号化サブルーチンを呼び出す。符号化サブルーチンへ渡す引き数は、ｏｆｓとｗｉｄｔｈである。以下、符号化サブルーチンの各ステップについて説明する。

【０１３３】符号化サブルーチンでは、予測を前半部系列と後半部系列に分けて行うため、予測の範囲を半分にする（ステップＳ６０）。すなわち、親ルーチンから引き数として受け取ったｗｉｄｔｈを１／２にする。そして、次のステップＳ６１で、前半部系列（配列のＡ〔ｏｆｓ〕からＡ〔ｏｆｓ＋ｗｉｄｔｈ－１〕まで）がすべて優勢シンボルか否かをチェックする。すべて優勢シンボルならステップＳ６２へ進む。ひとつでも劣勢シンボルが存在したら、直ちにステップＳ６４へ進む。

【０１３４】前半部系列がすべて優勢シンボルなら、符号語として“０”を出力する（ステップＳ６２）。そして、前半部系列の先頭位置を示すポインタｏｆｓにｗｉｄｔｈを加え、後半部系列の先頭位置を示すように変更する。また、前半部系列がすべて優勢シンボルのときは、後半部系列に必ず劣勢シンボルが存在するので、後半部系列の予測がはずれたことを示す符号語“１”を出力する必要がない。したがって、後述するステップＳ７０はスキップし、ステップＳ７１に進む。

【０１３５】一方、前半部系列に劣勢シンボルが存在する場合、符号語として“１”を出力する（ステップＳ６４）。次に、ｗｉｄｔｈが１を超えているか否かをチェックする（ステップＳ６５）。１以下の場合、これ以上分割できないので、子の符号化サブルーチン（ステップＳ６６）の呼び出しをスキップし、ステップＳ６７へ移行する。なお、ｗｉｄｔｈが２以上なら、さらに系列を２つに分け、それぞれを符号化しなければならない。そのための子の符号化サブルーチンを呼び出す（ステップＳ６６）。子の符号化サブルーチンは、図２２に示した符号化サブルーチンと全く同一となっている。つまり、ここでは、同一ルーチン（関数）の再帰呼び出しを行う。

【０１３６】符号化サブルーチンの再帰呼び出しによって前半部系列の符号化を終了すると、前半部系列の先頭位置を示すポインタｏｆｓにステップＳ６０で設定したｗｉｄｔｈを加え、後半部系列の先頭位置を示すように変更する（ステップＳ６７）。その後、後半部系列（配列のＡ〔ｏｆｓ〕からＡ〔ｏｆｓ＋ｗｉｄｔｈ－１〕まで）がすべて優勢シンボルか否かをチェックする（ステップＳ６８）。すべて優勢シンボルならステップＳ６９へ進む。ひとつでも劣勢シンボルが存在したら、直ちにステップＳ７０へ進む。そして、後半部系列がすべて優勢シンボルなら、符号語として“０”を出力する（ステップＳ６９）。

【０１３７】一方、前半部系列に劣勢シンボルが存在する場合、符号語として“１”を出力する（ステップＳ７０）。そして、次に、ｗｉｄｔｈが１を超えているか否かをチェックする（ステップＳ７１）。１以下の場合、これ以上分割できないので、子の符号化サブルーチンを実行するステップＳ７２をスキップし、次の注目系列の符号化工程へリターンする。なお、後半部系列についても、ｗｉｄｔｈが２以上なら、さらに系列を２つに分け、それぞれ符号化する。そのため図２２に示す符号化サブルーチンと同一の子の符号化サブルーチンを呼び出す（ステップＳ７２）。この符号化サブルーチンの再帰呼び出しによって後半部系列の符号化を実行する。

【０１３８】以上のような符号化プロセスの具体例を次に説明する。すなわち、符号化の具体例として、予測の初期値ｒｕｎを8、優勢シンボルを“0”として、“００００１００１”として表される入力ビットを符号化する場合について説明する。

【０１３９】まず、図２１の符号化メインルーチンのステップＳ５２で、Ａ〔0〕からＡ〔7〕に、上記の入力ビットを入力する。ステップＳ５３では、Ａ〔0〕からＡ〔7〕のすべてが“0”かどうか判定する。上の例の場合、ビット列に“1”が含まれているので、ステップＳ５５に移行し、まず符号語として“1”を出力する。続いてステップＳ５６では、ｗｉｄｔｈの大きさをチェックするが、ｗｉｄｔｈはこのとき8なので、符号化サブルーチン（ステップＳ５７）に進む。

【０１４０】符号化サブルーチンでは、まずステップＳ６０で、ｗｉｄｔｈを１／２の4に設定する。そしてステップＳ６１で、入力ビットの前半部、つまりＡ〔0〕からＡ〔3〕がすべて0かどうかチェックする。この場合、すべて“0”なのでステップＳ６２に進み、符号語として“0”を出力する。以上で前半部系列の符号化が完了する。続いてステップＳ６３を実行し、後半部系列の符号化に移るが、前半部系列がすべて“0”の場合、後半部系列に“1”が含まれるのは明らかである。したがって、ステップＳ７１でｗｉｄｔｈが1以下でない限り後半部系列をさらに分割して符号化しなければならない。そこで、符号化サブルーチンを子プロセスとしてステップＳ７２で再び呼び出す。なお、そのための前処理として、上述したようにステップＳ６３では、ｏｆｓにｗｉｄｔｈを加え、ｏｆｓを後半部系列の先頭位置にセットする。

【０１４１】ステップＳ７２では、ｏｆｓとｗｉｄｔｈを引き数として子の符号化サブルーチンを呼び出す。子の符号化サブルーチンを実行するステップＳ７２では、まず、図２２に示す符号化サブルーチンのステップＳ６０でｗｉｄｔｈをさらに半分にして2に変更する。次のステップＳ６１では、前半部系列、すなわちＡ〔4〕とＡ〔5〕が共に“0”であるか否かをチェックする。この場合、Ａ〔4〕が“1”なので、次のステップＳ６４に移行し、符号語として“1”を出力する。そしてステップＳ６５でｗｉｄｔｈが1を超えていると判断し、孫プロセスをステップＳ６６で呼び出す。孫の符号化サブルーチンでは、まずステップＳ６０においてｗｉｄｔｈが1となる。Ａ〔4〕は“1”なのでステップＳ６１からステップＳ６４へ処理が移り、符号語“1”を出力する。ステップＳ６５では、ｗｉｄｔｈが1以下なので、ステップＳ６６をスキップし、ステップＳ６７でｏｆｓを5に変更する。Ａ〔5〕は“0”なのでステップＳ６８からステップＳ６９に処理が移り、符号語“0”を出力する。

【０１４２】次に、この孫の符号化サブルーチンから抜けて、子の符号化サブルーチンのステップＳ６７に戻る。子の符号化サブルーチンのｏｆｓは4、ｗｉｄｔｈは2であるから、ステップＳ６７でｏｆｓは6に変更される。したがってステップＳ６８では、Ａ〔6〕とＡ〔7〕をチェックすることになる。この場合、Ａ〔7〕が“1”なのでステップＳ７０へ移行し、符号語“1”を出力する。そして、再び孫の符号化サブルーチンをステップＳ７２で呼び出す。孫の符号化サブルーチンでは、Ａ〔6〕が“0”なのでステップＳ６２で符号語“0”を出力する。そして、ｗｉｄｔｈが1なので、ステップＳ７２をスキップして子の符号化サブルーチンに復帰する。

【０１４３】子の符号化サブルーチンに復帰したプロセスは、さらに符号化メインルーチンに復帰し、ステップＳ５８で予測ｒｕｎの再設定と、優勢シンボルの再設定を行う。この例の場合、1次予測ははずれたが、2次予測で前半部が的中したので、ｒｕｎを8から4に変更し、優勢シンボルは引き続き“0”とする処理を施す。なお、予測ｒｕｎの設定は、2回続けてはずれたときに変更する等の設定にしても良い。

【０１４４】このような符号化プロセスによって、入力ビットである“００００１００１”が“１０１１０１０”の符号化系列となる。したがってこの場合、8ビットの入力系列が7ビットに圧縮されたことになる。

【０１４５】なお、復号化プロセスについては、符号化プロセスと逆のアルゴリズムによって、入力されてくる符号語を復号している。すなわち、復号化プロセスも、復号化メインルーチンと復号化サブルーチンにより構成され、符号化と逆のアルゴリズムによって復号している。

【０１４６】このように、図２１および図２２に示す符号化プロセスおよびその符号化プロセスと逆のアルゴリズムを使用して行う復号化プロセスでは、圧縮率が従来のＱＭコーダと呼ばれるものと同レベルであり、一方、符号化時間や復号化時間は大幅に短縮されたものとなっている。しかし、この符号化プロセスおよび復号化プロセスにおいては、予測ビット数であるｒｕｎで定まるビット列を一度に符号化しているため、圧縮率を高めるために、ｒｕｎの最大値を大きく設定すると、ｒｕｎの個数分のビットを保存しておくためのバッファが大きくなるという問題が生じる。

【０１４７】この問題は、マルコフ状態信号ＣＸをマルコフモデル化から得るような場合、そのバッファが非常に大きくなり、さらに大きな問題となる。すなわち、仮にｒｕｎをｎとしたとき、バッファとしてはｎビット分必要となり、さらにｍ状態のマルコフモデル化を行うと、バッファは各状態毎に必要となるため、ｎ×ｍビットの容量になる。この容量は、ｒｕｎの値が大きくなると無視できなくなる大きさとなる。

【０１４８】また、図２１および図２２に示す符号化プ

ロセスおよびその符号化プロセスと逆のアルゴリズムを使用する復号化プロセスでは、その各処理時間は、QMコーダに比べ大幅に短縮されているものの、符号化や復号化のサブルーチンを再帰的に呼び出して符号化や復号化を行っており、このサブルーチンの再帰的呼び出しのプロセスで時間を有するものとなっている。

【0149】このため、本実施の形態では、図18から図22に示す符号化プロセスおよび復号化プロセスを生かしつつ、デコード用のバッファを小さくしたり、符号化や復号化の時間をさらに減少できるデータ符号化方法等を採用している。以下、本発明の実施の形態におけるインデックス変換部7，8の動作の基本的な考え方を、図23から図34に基づき説明する。

【0150】まず、改良された本発明の実施の形態のエントロピー符号器12を、図23に基づき説明する。

【0151】このエントロピー符号器12は、エントロピー符号化装置となっており、符号化すべき2値ビット列を入力するビット列分解部22と、各予測ビット長run毎に符号化テーブルを内蔵する符号化テーブル部23と、符号化テーブル部23から入力される可変長符号を一旦バッファリングして固定のビット幅にならして出力するストリーム生成部24と、後述する状態遷移表を内蔵し、予測ビット長run等を設定する符号化制御部25とから主に構成される。

【0152】ここで、符号化制御部25には、状態遷移表を有する状態遷移部26と、マルコフモデル等により生成される符号化条件を入力し、その条件毎に現在の状態の信号を状態遷移部26に与え、符号化後に次の状態の信号入力し、その符号化の状態を記憶しておく状態記憶部27とが設けられている。

【0153】したがって、マルコフモデルのような条件付き符号化を行うときは、条件をインデックスとして、状態記憶部27から該当する状態を取り出し、その状態を後述する図28に示した状態遷移表により遷移させ、次の状態を再び状態記憶部27の元の番地にストアしておけば、条件毎に、状態を管理できることとなる。したがって、予測ビット長run等のパラメータも条件毎に個別に設定できることとなる。なお、マルコフモデル化する場合、ビット列分解部22には、各符号化条件毎に切り換えるバッファが複数必要となるが、この実施の形態では、予測ビット長runの数ではなく、より小さい固定の区切りビット数pで段階的に符号化しているので、そのバッファの容量はそれ程大きくならず、実用面で適したものとなっている。

【0154】ビット列分解部22は、符号化制御部25から予測ビット長runを指示する信号RUNと、優勢シンボルを指示する信号SWを入力する。ここで、信号RUNは、1からn（nは最大予測ビット長）の値を取る。また、信号SWは、その値が「0」のとき、“0”を優勢シンボルとし、「1」のとき、“1”を優勢シンボルとするが、その逆でも構わない。

【0155】さらに、ビット列分解部22は、デコードすべきビット数の信号DECNUMと、デコードすべきビットのパターンとなる信号パターンDECPATNを符号化テーブル部23に出力する。信号DECNUMは、入力ビット列に、劣勢シンボルを含む4ビットのパターンが現れたとき、その4ビットとそれまで続いた優勢シンボル個数の合計数となる。なお、信号RUNが「4」未満のときは、信号RUNと同じ値が出力される。これは、この実施の形態では、区切りビット数pを「4」としているためである。

【0156】このようにして、ビット列分解部22は、入力したビット列が信号RUNで指定されたビット数分、すべて信号SWで指定された優勢シンボルが続いたとき、すなわち、予測が的中したとき、信号DECNUMとして信号RUNの値を、信号パターンDECPATNとして“0”を出力する。

【0157】符号化テーブル部23は、図24から図27に示すような符号化テーブルを内蔵しており、どのテーブルを用いるかは、符号化制御部25からのテーブル番号指示信号TABLEにより選択される。そして、この符号化テーブル部23は、ビット列分解部22からの信号DECNUMと信号パターンDECPATNにより所定のテーブル内を検索し、所定の圧縮ビット列DECBITとそのビット長LENGTHおよび予測の当たり外れを示すFAILを出力する。なお、信号TABLEは、信号RUNと1対1の関係を有するものとなっている。

【0158】図24の符号化テーブルは、テーブル番号は「0」で、信号RUNの値が「1」、すなわちrunが「1」の場合を示している。図24に示されるように、runが「1」のときは、2種類の信号となっている。すなわち、デコードすべきビット数は1個であり、信号パターンは“0”と“1”の2種類となる。この2種類の入力信号に対して、圧縮ビット列DECBITと、そのビット長LENGTHと、予測の列外れを示すフラグFAILの組み合わせからなる2種類の信号が対応する。例えば、信号DECNUMが「1」で、信号パターンDECPATNが“0”の場合は、予測当たりとなり、フラグFAILは当たり信号の「0」となり、圧縮ビット列DECBITは“0”となり、ビット長LENGTHは「1」となる。

【0159】図25の符号化テーブルは、テーブル番号が「1」で、runが2の場合を示している。なお、各符号化テーブルの信号パターンDECPATNと圧縮ビット列DECBITは、共に右側から左側に入力してくる信号を示している。この図25の場合、その信号形態は4種類となる。デコードすべきビット数はすべて2個であり、そのときの信号パターンDECPATNは“00”“10”“01”“11”の4種類となる。信号パ

ターンＤＥＣＰＡＴＮが“００”のときは、２つとも優勢シンボルのため予測が当たったこととなり、フラグＦＡＩＬは当たり信号の「0」となると共に、そのときの圧縮ビット列ＤＥＣＢＩＴは“0”となり、ビット長ＬＥＮＧＴＨは「1」となる。

【０１６０】一方、信号パターンＤＥＣＰＡＴＮが“１０”のときは、劣勢シンボル“1”が入っており、予測が外れたこととなる。この結果、フラグＦＡＩＬは、外れ信号の「1」となり、圧縮ビット列ＤＥＣＢＩＴは、最初に“1”がくる。次に、“１０”の前半部が“0”であるため、予測が当たり圧縮ビット列ＤＥＣＢＩＴの２番目は“0”となり、“０１”となる。ここで、最初に予測外れとなっているので、後半部に“1”があることとなる。このため、圧縮ビット列ＤＥＣＢＩＴは、この“０１”がそのまま採用される。

【０１６１】信号パターンＤＥＣＰＡＴＮが“０１”のときは、劣勢シンボル“1”が入っており、予測が外れたこととなる。この結果、フラグＦＡＩＬは外れ信号の「1」となり、圧縮ビット列ＤＥＣＢＩＴは最初に“1”がくる。次に、“０１”の前半部が“1”であるため、予測がまたも外れたこととなり、圧縮ビット列ＤＥＣＢＩＴの２番目は“1”となる。信号パターンＤＥＣＰＡＴＮ“０１”の後半部は“0”であるため、予測当たりとなり、圧縮ビット列ＤＥＣＢＩＴの３番目は“0”となる。すなわち、信号パターンＤＥＣＰＡＴＮ“０１”に対応する圧縮ビット列ＤＥＣＢＩＴは、“０１１”となる。そして、ビット長ＬＥＮＧＴＨは「3」となる。同様にして、信号パターンＤＥＣＰＡＴＮ“１１”に対する圧縮ビット列ＤＥＣＢＩＴは、“１１１”となる。以上の４種類の信号の対応表が図２０となっている。

【０１６２】同様にして、テーブル番号が「2」で、ｒｕｎが「4」の１６種類の信号の対応関係が図２６に示され、テーブル番号「3」でｒｕｎが「8」の計３１種類の信号の対応関係が図２７に示されている。なお、図２６のｒｕｎが「4」の場合では、ｒｕｎの値が区切りビット数ｐと同じとなるので、図２４および図２５と全く同じ関係のみのものとなるが、図２７のｒｕｎが「8」の場合は、区切りビット数ｐ（この実施の形態ではｐ＝4）より大きくなるため、少し変更された表となる。

【０１６３】次に、他の表とは若干異なるこの図２７の符号化テーブルの内容を説明する。この符号化テーブルでは、デコードすべき信号のビット数ＤＥＣＮＵＭは、「8」のものと「4」のものが存在する。「8」のものは、前半部がすべて“００００”のものであり、「4」のものは、ｒｕｎが「8」で前半部に劣勢シンボル“1”がきた場合のものを示している。デコードすべき信号のビット数ＤＥＣＮＵＭ（以下単にＤＥＣＮＵＭとして示す）が「8」で、信号パターンＤＥＣＰＡＴＮ（以下単にＤＥＣＰＡＴＮとして示す）が“００００”のときは“００００００００”であることを示し、予測が当たったこととなり、フラグＦＡＩＬ（以下単にＦＡＩＬとして示す）は当たり信号の「0」となる。そして、圧縮ビット列ＤＥＣＢＩＴ（以下単にＤＥＣＢＩＴとして示す）は“0”で、ビット長ＬＥＮＧＴＨ（以下単にＬＥＮＧＴＨとして示す）は「1」となる。ＤＥＣＮＵＭが「8」で、ＤＥＣＰＡＴＮが“１０００”のときは、“１００００００”であることを示し、予測が外れたこととなり、ＦＡＩＬは外れ信号の「1」となる。そして、ＤＥＣＢＩＴの１番目には“1”がくる。次に、前半部“００００”は予測当たりとなり、ＤＥＣＢＩＴの２番目には“0”がくる。このとき、後半部“１０００”に劣勢シンボル“1”が当然くることとなるため、後半部の４つの信号に対するＤＥＣＢＩＴは、特に発生しない。

【０１６４】後半部“１０００”の中の前半部“００”は、予測当たりであり、３番目のＤＥＣＢＩＴは“0”となる。このとき、後半部“１０”に劣勢シンボル“1”が当然くることとなるため、後半部の２つの信号に対するＤＥＣＢＩＴは特に発生しない。そして、この後半部“１０”の前半部“0”は予測当たりとなり、４番目のＤＥＣＢＩＴは“0”となる。こうなると、最後尾に“1”があることが当然となり、特にＤＥＣＢＩＴは発生しない。よって、ＤＥＣＰＡＴＮ“１０００”に対応するＤＥＣＢＩＴは“０００１”となる。そして、ＬＥＮＧＴＨは「4」となる。これが、図２７のテーブル番号「3」の表の上から２番目の状態に対応する。

【０１６５】このような関係は、図２７の符号化テーブルの第３番目から第１６番目にも当てはまる。一方、図２７のテーブル番号「3」の上から第１７番目から第３１番目までは、ＤＥＣＮＵＭが「4」となり、図２１のテーブル番号「2」のものに近似する。すなわち、図２１の符号化テーブルの第２番目から第１６番目のものに、ｒｕｎが「8」として見たときの予測外れの“1”がすべて最初に付加されたものと、図２７のＤＥＣＮＵＭ「4」のものとは同一となる。なお、符号化テーブル部３より出力される符号は、ＬＥＮＧＴＨによって指定される可変長符号になっている。

【０１６６】この実施の形態では、さらにｒｕｎが１６のテーブルと、ｒｕｎが３２のテーブルとを有している。その両テーブルは、図２７に示すｒｕｎが8のテーブル番号「3」の考え方と同様となっており、その説明を省略する。

【０１６７】ストリーム生成部２４は、入力の可変長符号を一旦バッファリングして、出力の伝送路で定められた固定のビット幅にならして出力するものとなっている。

【０１６８】符号化制御部２５の基本動作は、信号ＲＵＮ（以下単にＲＵＮという）によってビット列分解部２

２にビットの切り出し方法を指示し、同時に信号ＴＡＢＬＥ（以下単にＴＡＢＬＥという）により符号化テーブルの選択を行うものとなる。そして、符号化テーブル部２３からフィードバックされるＦＡＩＬにより、次の符号化のためのＲＵＮとＴＡＢＬＥを設定する。なお、この実施の形態では、区切りビット数ｐを利用した段階的な符号化を導入したため、ある予測ビット長ｒｕｎで符号化した際、必要に応じて途中の段階であることをこの符号化制御部２５は記憶する必要がある。

【０１６９】この符号化制御部２５の具体的な動作は、図２８に示す状態遷移表に基づくものとなっている。この状態遷移表の動作について、予測当たりが続く場合を例にして説明する。ここで、初期状態は、ＳＳ１となっている。まず、状態ＳＳ１のとき、ｒｕｎが「１」で、ＴＡＢＬＥは「０」である。このため、図２４に示すテーブル番号「０」の符号化テーブルが使用される。そして、予測が当たる場合は、優勢シンボルが“０”が続くことであるため、入力されるビット列入力からそのＤＥＣＮＵＭの数である「１」個分の“０”のみを符号化テーブル部２３に送り、テーブル番号「０」のテーブル（＝図２４の表）に基づいて、ＦＡＩＬ「０」と、ＤＥＣＢＩＴ“０”と、ＬＥＮＧＴＨ「１」とが出力される。そして、そのＦＡＩＬ「０」が符号化制御部２５に伝えられる。

【０１７０】符号化制御部２５は、図２８の状態遷移表に基づき、ＳＳ１中のＦＡＩＬ「０」となるものを見つけ、次の状態として状態ＳＳ０を選択する（図２８の状態遷移表の上から３番目）。このとき、信号ＳＷは“０”となるので、シンボルの逆転はなく、そのまま“０”が優勢シンボルとなる。状態ＳＳ０においても、同様な動作の結果、状態遷移表の第１番目が選択され、状態ＳＳ３が次の状態となる。これによって、２回予測が当たったこととなる。

【０１７１】この状態遷移表では、２回予測が当たると、ｒｕｎが２倍になる。すなわち、上から７番目および８番の状態ＳＳ３となり、ｒｕｎが「２」となる。このように予測が当たり続けると、すなわち、入力ビット列がこの場合であると“０”であり続けると、ｒｕｎが「２」「２」「４」「４」「８」「８」と増えていく。また、一方、予測が外れ続けるときは、２回毎、同一ｒｕｎで行い小さくなっていく。すなわち、ｒｕｎが「８」「８」「６」「６」「４」「４」「２」「２」と小さくなっていく。そして、ｒｕｎが「１」のときに、予測が外れると、信号ＳＷは反転する。

【０１７２】このような状態遷移表の動作のルールをまとめると、次のとおりとなる。

【０１７３】(1)同一の予測ビット長ｒｕｎでの予測が２回連続して的中したとき、予測ビット長ｒｕｎを２倍する。

【０１７４】(2)同一の予測ビット長ｒｕｎでの予測が２回連続して外れたとき、予測ビット長ｒｕｎを１／２倍する。

【０１７５】(3)予測ビット長ｒｕｎが４以下のときは、１回で符号化を実行する。

【０１７６】(4)予測ビット長ｒｕｎが８で、ＤＥＣＮＵＭ＝４のときは、２回に分けて符号化を実行する。

【０１７７】(5)このときは、状態ＳＳ５に遷移して、予測ビット長ｒｕｎを「４」で、後半のビットを符号化する。

【０１７８】なお、信号ＳＷの反転とは、この値が１のとき、信号ＳＷを反転させるという意味である。

【０１７９】なお、図２８で示す状態遷移表は、ｒｕｎが「８」までしか示していないが、この実施の形態では、ｒｕｎを最大「３２」としているので、ｒｕｎ「１６」とｒｕｎ「３２」のものも、図示していないが同様に作成されている。また、状態遷移表としては、ｒｕｎが「６４」以上のものにしても良い。さらに、当たりや外れが２回続いたらｒｕｎを増加させたり減少させたりするのではなく、１回毎に変えたり３回以上の数としたり、種々のパターンを採用することができる。また、このような符号化テーブルとしては、ビット数の少ないものだけを用意し、大きなビット数、例えば、１６ビット以上の場合は符号化テーブルを持たないようにすることもできる。

【０１８０】次に、以上のような構成を有するエントロピー符号器１２の動作を具体例を使用して説明する。

【０１８１】例えば、予測が当たり続けて、ｒｕｎ＝１６となった状態で、“０００００１０００００１１１１０００……」のような形で入力してきたビット列を符号化する場合、４ビットの区切りビット数ｐで区切り、まず、最初は“０００００１００”までを符号化することとなる。これは、劣勢シンボル“１”が第１番目の区切りビット数（＝最初の４ビット）部分にはなく、第２番目（＝次の４ビット）に出てくるためである。そして、次に“００１１”を、そして最後に“１１００”を符号化することとなる。

【０１８２】このため、ビット列分解部２２から出力されるＤＥＣＮＵＭは、「８」「４」「４」となる。一方、ＤＥＣＰＡＴＮは、“０１００”“００１１”“１１００”（ここでは、いずれのパターンも左の数値から入力されてくるとする）となる。このような条件において、ＤＥＣＢＩＴは、まず、ｒｕｎ＝１６としたときの予測外れの“１”がくる。次に、“００００１００”は、ＲＵＮ「８」、ＴＡＢＬＥ「３」、ＤＥＣＮＵＭ「８」のため、図２７に示す上から５番目に相当するものであり（図２７に示す各数値の場合、それぞれ右端側から入力されてくることに注意）、ＤＥＣＢＩＴは“１０１００”となる。このため、先の“１”と合わせられた“１１０１００”（この数値は左端から順に出力）のＤＥＣＢＩＴとＬＥＮＧＴＨ「６」が符号化テーブル部

２３からストリーム生成部２４に出力される。

【０１８３】一方、符号化制御部２５内の状態遷移表でいえば、状態ＳＳ６でＤＥＣＮＵＭ「８」のとき、ＦＡＩＬ「１」となったこととなり、次の状態は状態ＳＳ７となる。そして、次の“００１１”は、ｒｕｎ＝８でＤＥＣＮＵＭ「４」なので、テーブル番号「３」のテーブル（＝図２７の符号化テーブル）が採用され、その上から１９番目のものが該当し、“１１０１１”のＤＥＣＮＵＭとＬＥＮＧＴＨ「５」が符号テーブル３から出力される。

【０１８４】最後の“１１００”については、前の状態が状態ＳＳ７のｒｕｎ「８」、ＤＥＣＮＵＭ「４」で、ＦＡＩＬ「１」となったため（図２８の現状態コード“０１１１”中の１番下の状態）、状態ＳＳ５が採用される。このため、ｒｕｎ「４」、ＴＡＢＬＥ「２」となり、図２６に示すテーブル番号「２」の符号化テーブルが使用される。そして、このテーブル番号「２」のテーブルにおいて、下から４番目が該当し“１１１１０”のＤＥＣＢＩＴと、ＬＥＮＧＴＨ「５」が符号化テーブル部２３からストリーム生成部２４に出力される。なお、状態ＳＳ５で、ＦＡＩＬは「１」となるので、次は状態ＳＳ２に移る。すなわち、次の入力ビット列に対しては、ｒｕｎ＝２である図２５の符号化テーブルが使用されることとなる。

【０１８５】以上をまとめると、入力ビット列“０００００１００００１１１００”が“１１０１００”，“１１０１１”，“１１１１０”の３つの圧縮ビット列として符号化されたこととなる。なお、入力ビット列や３つの圧縮ビット列は、共に先頭側から入力され、出力されていくものとする。この点、図２４から図２７の各符号化テーブルとは異なることに注意する必要がある。すなわち、各符号化テーブルでは、その表示の各値は、その表示の右端から順に入力し、出力するものとなっている。

【０１８６】そして、ｒｕｎは、当初「１６」であったのが、この４ビットの区切りビットｐで段階的に符号化していく中で、ｒｕｎは「２」となり、次の入力ビット列に対しては、「２」の予測ビット長ｒｕｎで符号化されることとなる。

【０１８７】一方、先に示した本実施の形態の元となる基本的プロセスで、同じ入力ビット列“０００００１０００００１１１００”を符号化すると、まずｒｕｎ＝１６での予測外れの“１”、次に前半の８ビットを注目し、２番目に予測外れの“１”がきて、さらに前半の４ビット“００００”に注目し、予測当たりの“０”が３番目にくる。すると、後半部の４ビット“０１００”に劣勢シンボルがくるとは確実なので、すぐに２つに分割し、前半の２ビット“０１”に注目する。このため、予測外れの“１”が４番目にくる。次は、さらにこれを２分割し、前半の“０”に注目し、５番目に予測当たりの“０”がくる。すると、後半の“１”は劣勢シンボルが確実なので、すぐに後半の２ビット“００”に注目し、予測当たりの“０”が６番目にくる。

【０１８８】以上の前半８ビットの符号化をまとめると、“１１０１００”となる。これは、本実施の形態による符号化ビットと全く同じとなる。続く８ビットも同様な方法で進めていくと、これらも本実施の形態による符号化ビットと同一となる。本実施の形態の元となる基本的プロセスと本実施の形態とが異なる点は、符号化されたビット自体ではなく、1.符号化の区切り方、2.予測ビット表の変更の仕方、3.符号化テーブルの活用の３点にある。

【０１８９】すなわち、改良した本実施の形態では、入力ビット列に対しｒｕｎより小さい区切りビット数ｐ（この実施の形態ではｐ＝４）で区切り、劣勢シンボルが存在する区切り部分までで一旦符号化を区切るようにしている。先の例では、１６ビットの入力ビット列が３つに区切られて符号化されている。また、本実施の形態では、次の入力ビット列に対し予測ビット長ｒｕｎは「２」となるのに対し、基本的プロセスの考え方では、予測外れは１回であり、ｒｕｎは「１６」のままとなる。さらに、本実施の形態の基本的プロセスの考え方では、符号化サブルーチンを再帰的に呼び出して符号化しているが、改良した本実施の形態の符号化方法では、符号化テーブル、具体的には予測ビット長ｒｕｎ毎に符号化テーブルを用いている。

【０１９０】以上の３つの点は、それらが同時に利用されることによって大きな効果を生ずるが、それぞれ単独で使用されても十分効果を有する。例えば、第１の点の段階的に符号化する方法を採用すると、バッファ、例えば、ビット列分解部２２やストリーム生成部２４内の各バッファを小さくできるばかりか後述するマルコフモデル化によって圧縮ビット列を得ようとするときにそのバッファの容量を減少させることができる。

【０１９１】第２の点の予測ビット長ｒｕｎの変更については、入力ビット列が途中からがらっとその性質が変わるような場合に特に有効となる。先の例では、予測が当たり続けてｒｕｎ＝１６となったのに対し、次に性質ががらっと変わったビット列、すなわち劣勢シンボルを多く含む“０００００１００００１１１００”がきたとき、改良された本実施の形態の符号化方法では、その性質に合わせ、ｒｕｎは「２」となり、続く入力ビット列の性質に合う確率が高いものとなり、圧縮率が高くなる。しかし、基本的プロセスで処理した場合、ｒｕｎは「１６」のままであり、次の入力ビット列の性質にそぐわない確率の高いものとなる。なお、圧縮率の向上は、具体的には、改良前に比べ、０．５％から数％程度であるが、各プログラムソフト等が大容量化している現在では、このようなわずかな数値の向上効果も無視し得ないものとなっている。

【０１９２】第３の点の符号化テーブルについては、サブルーチンの再帰的呼び出しによる符号化に比べ、符号化テーブルのためのメモリ容量は若干増えるものの、符号化速度が極めて速くなる。

【０１９３】次に、この実施の形態のエントロピー復号器３２について、図２９に基づき説明する。

【０１９４】このエントロピー復号器３２は、符号化された信号のストリームを入力するストリーム切り出し部４１と、予測ビット長ｒｕｎに応じた複数の復号テーブルを内蔵する復号テーブル部４２と、復号されたビットをストアし、所定のシンボルを出力するデコードバッファ部４３と、エントロピー符号器１２の状態変換部２６内の状態遷移表と同じ状態遷移表他を有する復号制御部４４とから主に構成されている。ここで、復号制御部４４には、状態遷移表を有する状態遷移部４５と、マルコフモデル等により生成される復号条件を入力し、その条件毎に現在の状態の信号を状態遷移部４５に与え、復号後に次の状態の信号を入力し、その復号条件の状態を記憶しておく状態記憶部４６とが設けられている。

【０１９５】なお、デコードバッファ部４３は、復号条件が入力し、その条件毎に個別に管理されるものとなっている。このため、マルコフモデルのような条件付き復号化の場合、バッファとして非常に大きなものが必要になる。しかし、本実施の形態のエントロピー復号器３２では、後述するように段階的な復号を行うので、各バッファは小さいものでも十分対応でき、マルコフモデルのような条件付きの復号化でもデコードバッファ部４３はそれ程大きな容量を必要としなくなる。

【０１９６】ストリーム切り出し部４１は、復号テーブル部４２から、復号したビット数を後述するＬＥＮＧＴＨにより指示されるので、その値に基づき、復号済みビットを廃棄して、未復号ビットの先頭が、符号化されたデータとなる復号予定の符号語信号ＣＯＤＥ（以下単にＣＯＤＥという）の最下位ビット（または最上位）に来るようにストリームを切り出す。なお、ＬＥＮＧＴＨを評価して、復号済みビットを廃棄するのは、デコードバッファ部４３から廃棄指示ＤＥＣＲＥＱ（以下単にＤＥＣＲＥＱという）があったときのみである。また、ＣＯＤＥは８ビット単位で送信される。

【０１９７】復号テーブル部４２は、図３０から図３３に示すような各復号テーブルを内蔵し、復号制御部４４が出力するテーブル番号指示信号ＴＡＢＬＥ（以下単にＴＡＢＬＥという）によりそれらを切り替えて使用する。そして、復号テーブル部４２は、次の信号を出力する。すなわち、(1)何ビット復号したかを示す信号ＬＥＮＧＴＨ（以下単にＬＥＮＧＴＨという）で、エントロピー符号器１２におけるＬＥＮＧＴＨに相当するもの、(2)予測の当たり外れを示す信号ＦＡＩＬ（以下単にＦＡＩＬという）で、エントロピー符号器１２におけるＦＡＩＬに相当するもの、(3)復号したビット・パターン信号ＤＥＣＰＡＴＮ（以下単にＤＥＣＰＡＴＮという）で、エントロピー符号器１２におけるＤＥＣＰＡＴＮに相当するもの、(4)復号結果が何ビットかを示す信号ＤＥＣＮＵＭ（以下単にＤＥＣＮＵＭという）で、エントロピー符号器におけるＤＥＣＮＵＭに相当するもの、を出力する。

【０１９８】図３０に示すｒｕｎ＝１の復号テーブルは、ＣＯＤＥが“０”“１”の２種類に対応する各出力が記載されている。この復号テーブルは、図２４のｒｕｎ＝１の符号化テーブルに相当するもので、符号化テーブル中のＤＥＣＢＩＴに相当するものが、この復号テーブルではＣＯＤＥとなっている。図３１に示すｒｕｎ＝２の復号テーブルは、同様に図２５のｒｕｎ＝２の符号化テーブルに相当するものとなっている。また、図３２に示すｒｕｎ＝４の復号テーブルでは、図２６のｒｕｎ＝４の符号化テーブルに相当し、図３３に示すｒｕｎ＝８の復号テーブルは、図２７のｒｕｎ＝８の符号化テーブルに相当している。なお、各復号テーブルにおける各数値も、符号化テーブルと同様に、各数値の右端側から入力し、出力する表示となっている。また、復号テーブルとして、ｒｕｎ＝１６と、ｒｕｎ＝３２のテーブルも用意されている。

【０１９９】デコードバッファ部４３は、４ビット（この実施例の場合）以下のＤＥＣＰＡＴＮとＤＥＣＮＵＭを直接的にストアし、それぞれデコードバッファ部４３内のＰＡＴＮＲＥＧ（以下単にＰＡＴＮＲＥＧという）とナンバーレジスタＮＵＭＲＥＧ（以下単にＮＵＭＲＥＧという）にストアする。そして、デコードバッファ部４３の出力がｑビット幅の場合、デコードバッファ部４３は、１回デコード・データを出力する度にストアしたＮＵＭＲＥＧからｑを減じる。そして、ＮＵＭＲＥＧがｑより小さくなったら、ＤＥＣＲＥＱをアクティブにして、新たなデータのデコード要求を発する。また、ＮＵＭＲＥＧが５以上のときは、信号ＳＷで定まる優勢シンボルをデコード出力として出力する。一方、ＮＵＭＲＥＧが４以下になったら、ＰＡＴＲＥＧの値を出力する。

【０２００】例えば、図３３の上から５番目のＣＯＤＥ“００１０１”が復号テーブル部４２に入力された場合、ＤＥＣＮＵＭ＝８、ＤＥＣＰＡＴＮ＝“００１０”がデコードバッファ部４３に入力されてくる。このとき、信号ＳＷが「０」となっていたとし、出力を２ビット単位（これはｑ＝２に相当）で行うとした場合、最初の２回の出力は優勢シンボルを出力すればよい。この場合ＳＷ＝０なので、優勢シンボルは“０”である。したがって、“００００”を出力する。この４ビットを出力した時点で、ＮＵＭＲＥＧは４（＝８－４）になっている。そこで、次のサイクルは、ＰＡＴＮＲＥＧの値を、順に出力する。すなわち、“０１００”をこの表示の左端側から出力する。

【０２０１】復号制御部４４の状態遷移部４５は、符号

化制御部２５の状態遷移部２６と同じ状態遷移表を保有している。そして、状態の初期値は、ＳＳ１であり、ＦＡＩＬとＤＥＣＮＵＭにより、次の遷移先が決定され、ＤＥＣＲＥＱがアクティブのとき、その遷移先へ遷移する。

【０２０２】以上のように構成されるエントロピー復号器３２は、先に示したエントロピー符号器１２と逆のアルゴリズムによって動作する。なお、このエントロピー復号器３２は、デコードバッファ部４３の出力状態によって制御されるものとなっている。すなわち、デコードバッファ部４３のＮＵＭＲＥＧが出力ビット幅ｑより小さくなると、ＤＥＣＲＥＱがストリーム切り出し部４１と復号制御部４４へ出力される。ストリーム切り出し部４１は、そのＤＥＣＲＥＱにより復号済みビットをそのＬＥＮＧＴＨ分廃棄する。

【０２０３】先の例のｒｕｎ＝８でＣＯＤＥ“００１０１”の場合、ＮＵＭＲＥＧが「８」から「４」へ、「４」から「２」、「２」から「０」へと下がる。この「２」から「０」へ下がったときに、ＤＥＣＲＥＱが発生する。そして、ＬＥＮＧＴＨが「５」であるので、ＣＯＤＥから復号済みの５ビットを廃棄する。このため、ストリーム切り出し部４１内のＣＯＤＥには、未復号ビットが最下位または最上位にきて、次の復号に備える。一方、復号制御部４４では、ｒｕｎ＝８、ＤＥＣＮＵＭ＝８で、ＦＡＩＬ＝１なので、状態ＳＳ７へ遷移する。このため、ｒｕｎ＝８に相当するＴＡＢＬＥ＝３を復号テーブル部４２に向けて出力する。

【０２０４】この結果、復号テーブル部４２は、図３３のテーブル番号「３」であるｒｕｎ＝８の復号テーブルを準備する。そして、入力してくるＣＯＤＥからＬＥＮＧＴＨ、ＤＥＣＮＵＭ、ＤＥＣＰＡＴＮおよびＦＡＩＬが確定し、出力される。例えば、そのＣＯＤＥの最初が“０”であれば、ＣＯＤＥ“０”であることが確定し、ＬＥＮＧＴＨ＝１、ＤＥＣＮＵＭ＝８、ＤＥＣＰＡＴＮ＝“００００”、ＦＡＩＬ＝「０」を出力する。一方、ＣＯＤＥが“０１０１１”の場合、ＣＯＤＥの最初が“１”であるので、まだ確定せず、次の“１”でも、３番目の“０”でも、４番目の“１”でも確定しない。しかし、５番目の“０”が入った段階で“０１０１１”であることが確定する。この確定によって、ＬＥＮＧＴＨ＝５、ＤＥＣＮＵＭ＝４、ＤＥＣＰＡＴＮ＝“０１００”、ＦＡＩＬ＝「１」がそれぞれ出力される。このようにして、順次、復号されていく。

【０２０５】このエントロピー復号器３２は、エントロピー符号器１２と同様に、先に述べた基本的プロセスに基づく復号に比べると、①段階的な復号によるバッファ容量の減少化②信号の性質にあった予測ビット長ｒｕｎの変更③復号テーブルによる復号速度の向上という各種の有利な効果を有するものとなる。

【０２０６】このエントロピー復号器３２においては、状態遷移を条件毎に個別に管理する点で、エントロピー符号器１２と同様である。ただし、このエントロピー復号器３２の場合は、上述したようにさらにデコードバッファ部４３も個別に管理しなければならない。このため、デコードバッファ部４３は、ＮＵＭＲＥＧ、ＰＡＴＮＲＥＧに相当するレジスタを有り得る条件数分内蔵し、復号条件によって切り換えるものとなっている。

【０２０７】上述の説明では、分かり易さを考慮し、２値のビット列を例にして、予測ランレングス符号化方式を説明したが、判別部１９から出力される順位コードは多値データであり、実際は多値データを２値系列に変換する必要がある。例えば、ビット・プレーンに分けて、各ビット・プレーンをこの予測ランレングス符号化方式で符号化するようにしても良い。また、最上位ビットからプレーン毎にこの予測ランレングス符号化方式にて符号化を行い、“１”が出現した時点で続く下位ビットを直接ストリームに出力するようにしても良い。

【０２０８】この実施の形態では、このＰＲＬＣを多値系列に適用する方式として、ビット・プレーンではなくレベル・プレーンに分けて行っている。具体的には、シンボルが８ビットであるため、２５６のレベル・プレーンに分け、その入力シンボルをグループに分け、グループ番号をこの予測ランレングス符号化方式で符号化している。すなわち、入力シンボルを図３４に示すように、グループ分けし、まず入力シンボルがグループ番号０か０以外かを示す判定ビットをこのＰＲＬＣで符号化する。もし入力シンボルが０ならこのシンボルの符号化を完了するが、そうでない場合はさらにグループ番号が１か１以外かを示す判定ビットをこの予測ランレングス符号化方式で符号化する。

【０２０９】このようにして、グループ番号が確定するまで、判定ビットをＰＲＬＣで符号化し、確定したグループ番号が２以上の場合は、該当グループにおけるシンボルを確定するため、必要とする付加ビットを直接ストリームに出力する。ただし、各グループ判定ビットは、グループ毎に独立した系列として扱い、各々個別に現状態コードを管理して符号化する。この方法は、グループ番号が確定した時点で、上位の判定ビットの符号化を行わないので、処理速度が向上する。

【０２１０】なお、この実施の形態では、圧縮対象画素の画素インデックス３の数が２５６個に加え、（ＣＸ＋１）が加わるため、最高２６０個が対象となり、順位としては２５９順位となる。この対策として、順位２５５をエスケープ・コードに割り当て、２５５以上の順位については、このエスケープ・コードを符号化後、２５５との差分を３ビット付加ビットとして直接圧縮ストリームに出力している。このような対策は、圧縮対象の画素インデックス３が２５６色をわずかに超える場合にも適用できる。

【０２１１】また、この実施の形態では、動画像のイン

デックスパレットは、動画ストリームの中で共通のパレットとして使用されている。さらに、このパレットは、フレーム間相関を用いた符号化では、フレーム間相関の及ぶ範囲で共通のパレットを使用している。例えば、Ｉピクチャーとピクチャーの間（ＧＯＰ）毎にパレットを持つようにしたり、他の範囲で持つようにしたりしている。

【０２１２】また、この実施に形態のエントロピー符号器１２では、符号化する際、過去の符号化済み系列等により適当に条件付けを行い、符号化することで、圧縮率が向上している。すなわち、マルコフ状態信号ＣＸにより条件分けを行ってＰＲＬＣ符号化を実行している。具体的には、マルコフ状態信号ＣＸは、４状態存在し、１つの状態で８つのグループ判定ビットを個別に符号化するため、合計で３２状態の現状態コードを個別に管理し符号化することになる。

【０２１３】なお、上述の実施の形態は、本発明の好適な実施の形態の例であるが、これに限定されるものではなく本発明の要旨を逸脱しない範囲において、種々変形実施可能である。例えば、カラー画素データ１００Ａとしては、ｎビット（ｎは２以上の整数）のカラー画素データ１００Ａを対象とし、その符号化や復号化を行うように構成することができる。

【０２１４】なお、符号化や復号化を行うにあたり、図１８から図２２に示す改良前の方法を採用したり、図４０に示す従来の算術符号型のエントロピー符号器およびエントロピー復号器を利用しても良い。また、上述の実施の形態のように、プリスキャンを行うのではなく、特開平６－２７６０４１号に開示された技術を利用しても良い。すなわち、インデックスを出現頻度順に並び替えた色順位データを利用したり、最新のものを先頭へ移動する移動処理を行って得た最新出現表を利用するようにしても良い。

【０２１５】さらに、先頭移動処理を使用する場合、現れたカラーシンボルを常に先頭へ持っていくのではなく、先頭方向へ所定の決まりにしたがって移動させるようにしても良い。このようにすると、出現確率が低いカラーシンボルが出たとき、すぐに先頭へ移動しないので、出現確率の低いものに短い符号を割り当てるということが無くなる。このため、出現確率が低いものが現れても符号化効率や復号化効率が悪化してしまうことはない。

【０２１６】なお、マクロブロックＭＢと小さなブロックＳＢによる二重判定は、マルチカラー画像以外の画像、例えば自然画の動き補償の際にも適用できる。また、動きベクトルを利用しないとしたマクロブロックＭＢの拾い方は、フレームのスキャンが横方向のときは縦方向に集めていき、フレームのスキャンが縦方向のときは横方向に集めていくのが好ましい。

【０２１７】また、動き補償を行うブロックは、一般的に採用されている１６×１６ピクセルの他に８×８ピクセル、４×４ピクセル、３２×３２ピクセル等種々の大きさのものとすることができるが、動きベクトルのためのデータ量を考慮すると４×４ピクセル以上が好ましく、効果的な動き検出を考慮すると６４×６４ピクセル以下が好ましい。

【０２１８】さらに、上述の実施の形態では、符号化および復号化は、ハードとして構成されたシステムで処理しているが、すべてソフトウェアで処理するようにしても良い。また、ソフトウェアで処理する際は、その処理手順をプログラム化しＣＤ－ＲＯＭ等の記憶媒体にインストールし、その記憶媒体をパソコン等のハードに差し込んで処理したり、そのプログラムをネットワーク等を介して送受信し、パソコン等のハードディスク等へ取り込んで処理するようにしても良い。

【０２１９】

【発明の効果】以上説明したように、本発明のマルチカラー画像からなる動画像の符号化装置およびその方法では、マルチカラー画像にも動き補償を適切に利用できるようになり効果的なデータ圧縮が可能となる。

【０２２０】また、本発明のマルチカラー画像からなる動画像の復号化装置およびその方法では、マルチカラー画像の復号化にも動き補償を利用でき、復号化装置のメモリの低容量化が可能となると共に復号処理を高速化することができる。

【図面の簡単な説明】

【図１】本発明のマルチカラー画像からなる動画像の符号化装置およびその方法を採用した符号化システムを示す図である。

【図２】図１の符号化システム中のインデックス変換部およびその周辺の構成を示す図である。

【図３】本発明のマルチカラー画像からなる動画像の復号化装置およびその方法を採用した復号化システムを示す図である。

【図４】図３の復号化システム中のインデックス変換部およびその周辺の構成を示す図である。

【図５】図１の符号化システムの動き補償部の動作結果を説明するための図で、（Ａ）は各ピクチャー（フレーム）の種類と入力順序を示す図で、（Ｂ）はそのときの符号化順序を示す図である。

【図６】図１の符号化システムおよび図３の復号化システムで採用されるマクロブロックと小さなブロックおよびその関係を示す図である。

【図７】図１および図３の符号化システムおよび復号化システムにマルコフモデルとして採用される参照画素の配置を説明するための図で、（Ａ）は通常の参照画素を、（Ｂ）は先頭ラスタの場合を、（Ｃ）は先頭画素の場合をそれぞれ示している図である。

【図８】図１および図３の符号化システムおよび復号化システムに採用される参照画素の状態と、状態信号ＳＴ

と、色数と、マルコフ状態信号ＣＸとの関係を示す図である。

【図９】図１の符号化システムに採用される変換テーブル生成部の参照順位テーブル作成の機能を説明するための図で、（Ａ）は符号化対象画素と参照画素が一致したときの度数の例を示す図で、（Ｂ）（Ｃ）はそれぞれの度数が所定の関係となったときの参照画素の順位を示す図である。

【図１０】図１および図３の符号化システムおよび復号化システムに採用される参照順位テーブルの例を示す図である。

【図１１】図１の符号化システムに採用される変換テーブル生成部の画素インデックス変換テーブル作成機能を説明するための図で、入力してきた画素インデックスに対応する画素が参照画素中に無かった度数を示す図である。

【図１２】図１の符号化システムに採用される変換テーブル生成部の画素インデックス変換テーブル作成機能を説明するための図で、入力してきた画素インデックスに対応する画素が参照画素中に無かった度数の順位を示す図である。

【図１３】図１の符号化システムにおける入力してきたカラーパレットから変換カラーパレットへの変換を説明するための図である。

【図１４】図１の符号化システムにおけるプリスキャン時の動作を示すフローチャートである。

【図１５】図１の符号化システムにおける符号化プロセス時の動作を示すフローチャートである。

【図１６】図１の符号化システムにおいて動き補償を行わないと判断されたマクロブロックの符号化方法を説明するための図で、（Ａ）は動き補償を行わないマクロブロックの位置の例を示し、（Ｂ）はその一部拡大図である。

【図１７】図３の復号化システムにおける復号化プロセスを示すフローチャートである。

【図１８】本発明で採用するエントロピー符号器およびエントロピー復号器で使用しているアルゴリズムの概要を説明するための図で、注目系列と予測ビット数ｒｕｎとの関係を示す図である。

【図１９】図１８の注目系列を分割した状態を示す図である。

【図２０】図１９の前半部注目系列をさらに分割した状態を示す図である。

【図２１】図１８から図２０に示すエントロピー符号器内の符号化プロセスを説明するためのフローチャートで、符号化メインルーチンを示すフローチャートである。

【図２２】図１８から図２０に示すエントロピー符号器内の符号化プロセスを説明するためのフローチャートで、符号化サブルーチンを示すフローチャートである。

【図２３】本発明の実施の形態で採用するエントロピー符号器の構成を示すブロック図である。

【図２４】図２３のエントロピー符号器の符号化テーブル部内の符号化テーブルを示す図で、予測ビット長が「１」の場合のテーブルを示す図である。

【図２５】図２３のエントロピー符号器の符号化テーブル部内の符号化テーブルを示す図で、予測ビット長が「２」の場合のテーブルを示す図である。

【図２６】図２３のエントロピー符号器の符号化テーブル部内の符号化テーブルを示す図で、予測ビット長が「４」の場合のテーブルを示す図である。

【図２７】図２３のエントロピー符号器の符号化テーブル部内の符号化テーブルを示す図で、予測ビット長が「８」の場合のテーブルを示す図である。

【図２８】図２３のエントロピー符号器の状態遷移部内の状態遷移表を示す図である。

【図２９】本発明の実施の形態で採用するエントロピー復号器の構成を示すブロック図である。

【図３０】図２９のエントロピー復号器の復号テーブル部内の復号テーブルを示す図で、予測ビット長が「１」の場合のテーブルを示す図である。

【図３１】図２９のエントロピー復号器の復号テーブル部内の復号テーブルを示す図で、予測ビット長が「２」の場合のテーブルを示す図である。

【図３２】図２９のエントロピー復号器の復号テーブル部内の復号テーブルを示す図で、予測ビット長が「４」の場合のテーブルを示す図である。

【図３３】図２９のエントロピー復号器の復号テーブル部内の復号テーブルを示す図で、予測ビット長が「８」の場合のテーブルを示す図である。

【図３４】本発明で採用するアルゴリズムを本実施の形態で示す多値系列データに適用した場合の例を説明するための図で、入力シンボルを複数のグループに分けた状態を示す図である。

【図３５】従来のマルチカラー画像の符号化システムおよび復号化システムのブロック図である。

【図３６】従来の符号化対象画素データに対する参照画素データの説明図である。

【図３７】従来のパラメータテーブルを示す図である。

【図３８】状態縮退器を有する従来のマルチカラー画像の符号化システムおよび復号化システムのブロック図である。

【図３９】従来の縮退テーブルの一例を示す図である。

【図４０】従来の算術符号型のエントロピー符号器およびエントロピー復号器の説明図である。

【図４１】従来や本発明で使用されるマルチカラー画像のインデックスを説明するための図である。

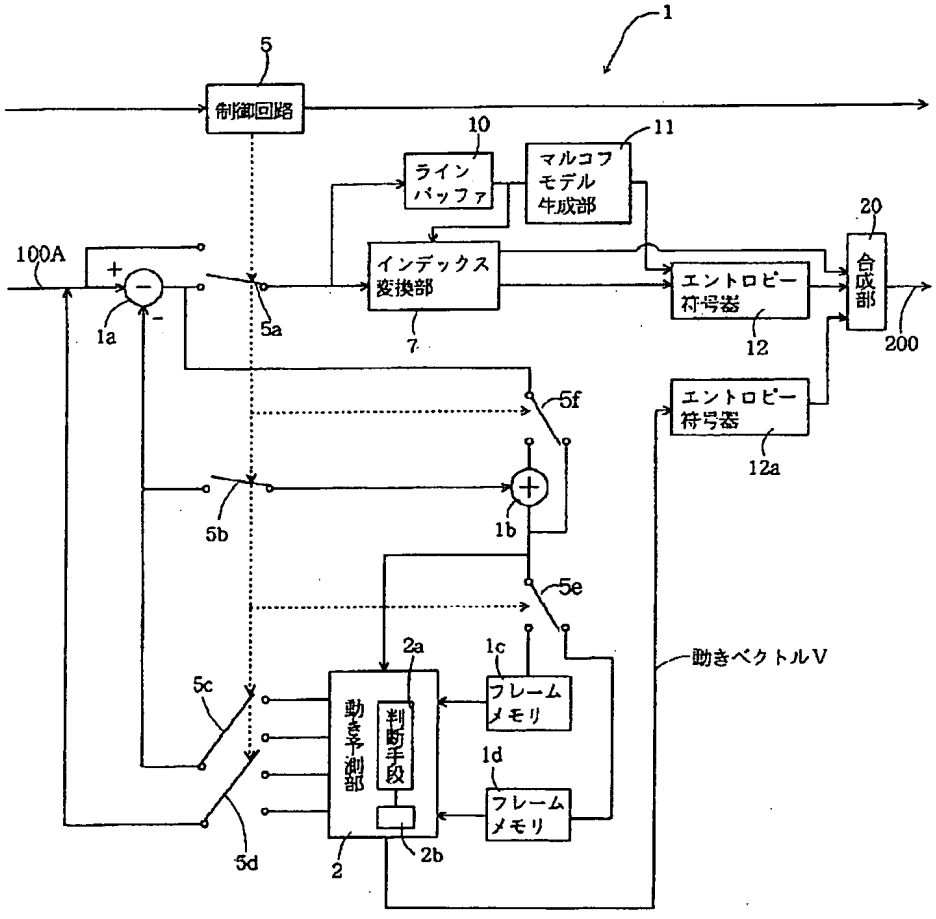
【図４２】従来の合成された色順位テーブルの作成原理を示す説明図で、（Ａ）は各画素の配置関係を示し、（Ｂ）は各画素のカラーシンボルを示し、（Ｃ）は合成

された色順位テーブル（最新出現表）を示す図である。

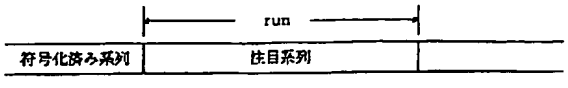
【符号の説明】

１　符号化システム
２　動き予測部
２ａ　判断手段
３　復号化システム
４　動き補償部
４ａ　判断手段
７、８　インデックス変換部
１０　ラインバッファ（周辺画素生成手段）
１１　マルコフモデル作成部（縮退手段）
＊１２、１２ａ　エントロピー符号器（エントロピー符号化手段）
２０　合成部
３０　ラインバッファ
３１　マルコフモデル生成部（縮退手段）
３２、３２ａ　エントロピー復号器（エントロピー復号化手段）
３３　分離部
１００Ａ　符号化されるカラー画素データ
１００Ｂ　復号化されたカラー画素データ
＊　２００　符号化データ

【図１】

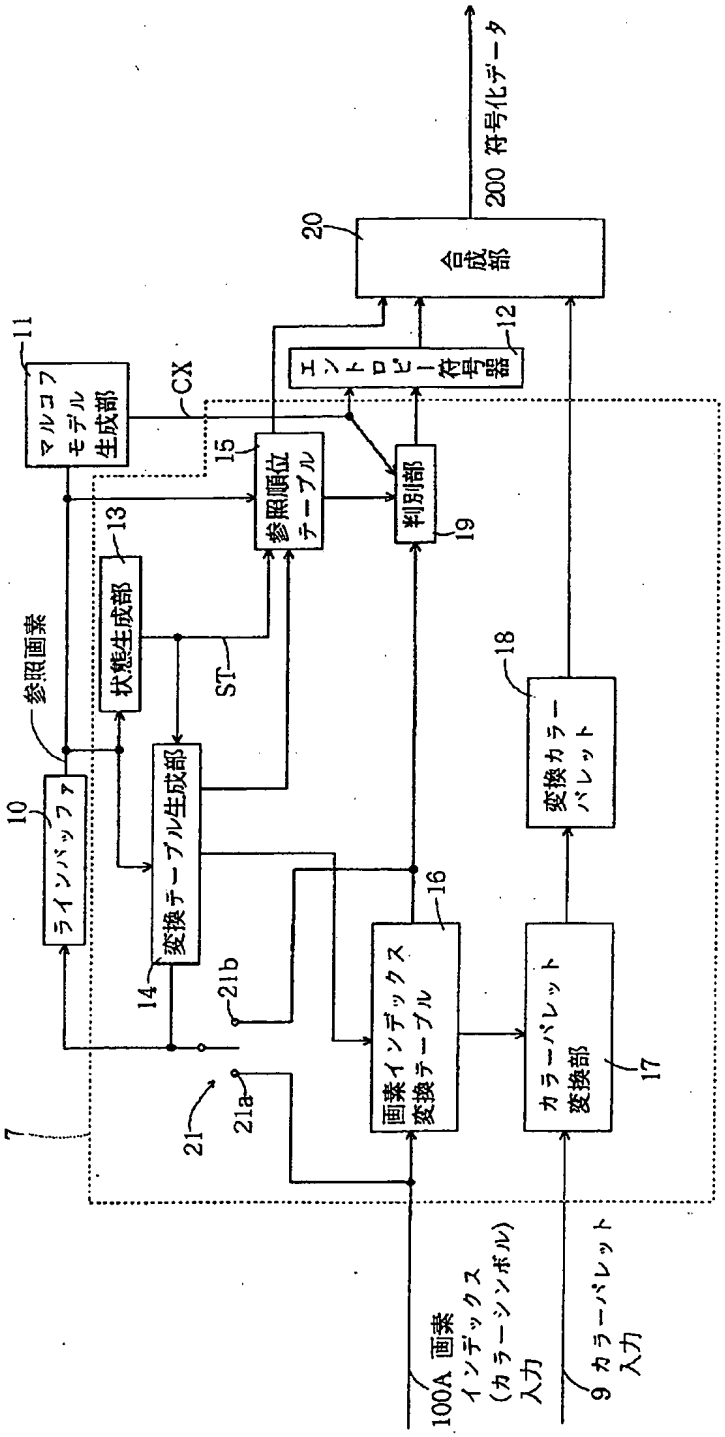


【図１８】

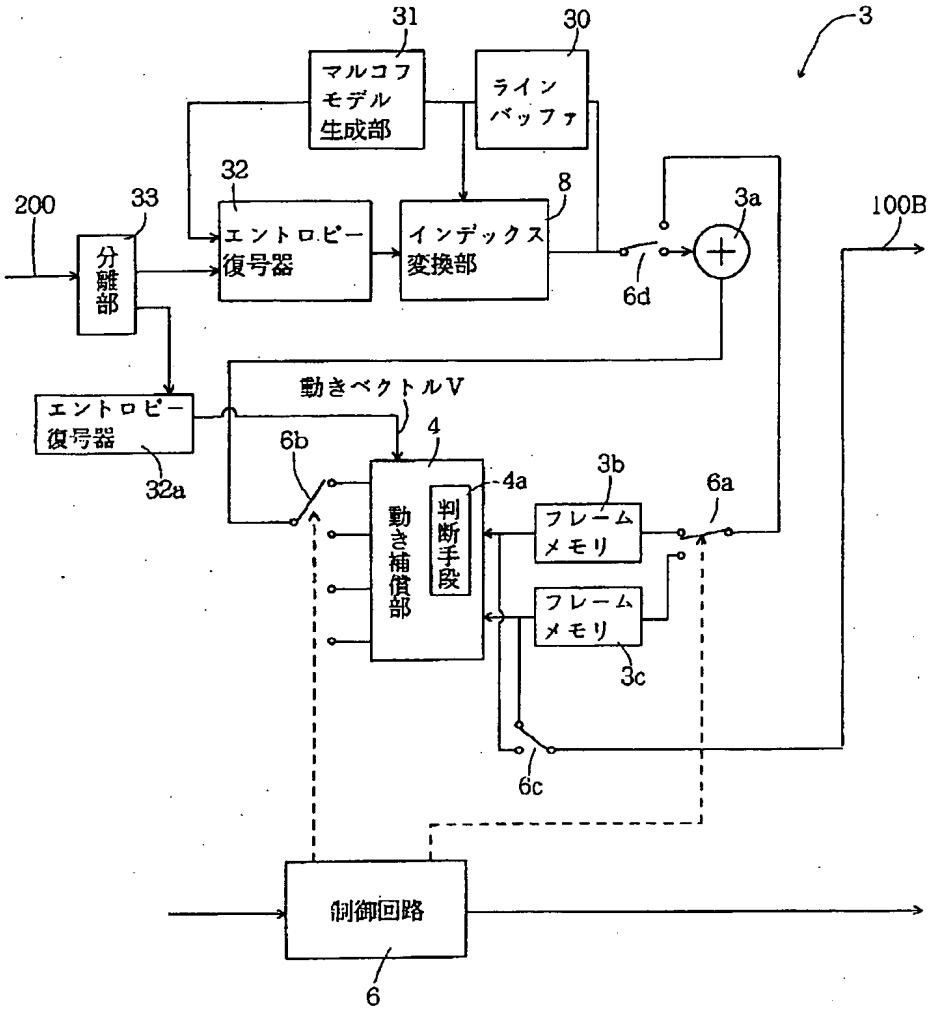


【図１９】

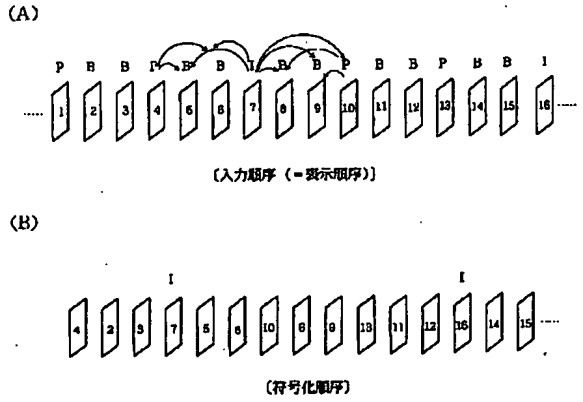
【図２】


200 符号化データ
20
合成部
エントロピー符号器
12
11
マルコフモデル生成部
CX
15
参照順位テーブル
判別部
19
13
状態生成部
ST
参照画素
10
ラインバッファ
変換テーブル生成部
14
21b
21
21a
7
画素インデックス変換テーブル
16
カラーパレット変換部
17
18
変換カラーパレット
100A 画素インデックス（カラーシンボル）入力
9 カラーパレット入力

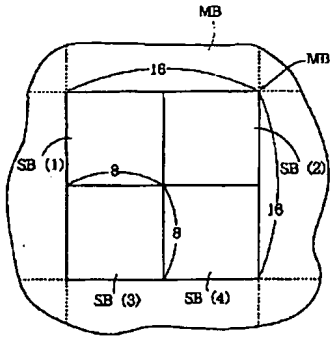
【図３】


31
30
3
マルコフ
モデル
生成部
ライン
バッファ
32
8
200
33
3a
100B
エントロピー
復号器
インデックス
変換部
分離部
6d
動きベクトルV
エントロピー
復号器
4
6b
3b
6a
4a
判断手段
動き補償部
フレーム
メモリ
フレーム
メモリ
32a
3c
6c
制御回路
6

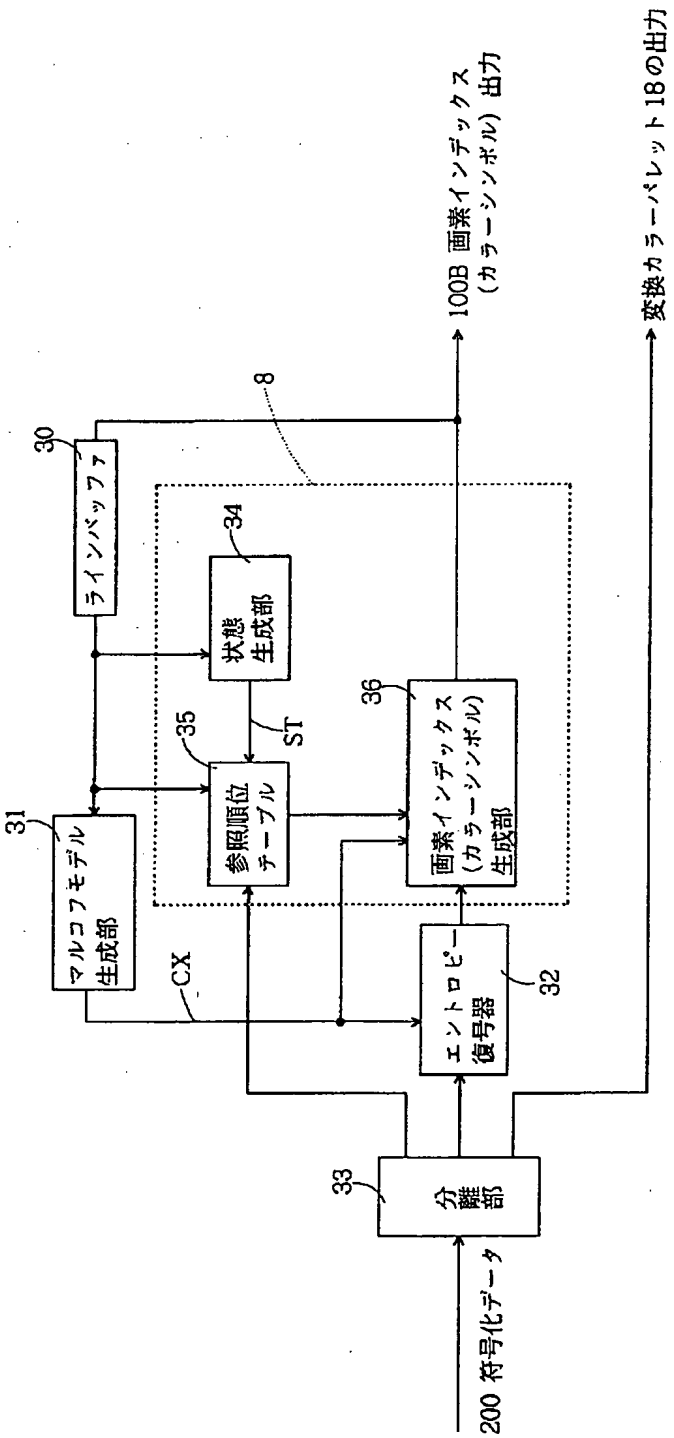
【図５】


(A)
P B B P B B I B B P B B P B B I
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
〔入力順序（＝表示順序）〕
(B)
I I
4 2 3 7 5 6 10 8 9 13 11 12 16 14 15
〔符号化順序〕

【図６】


MB
16
MB
SB (1)
8
SB (2)
16
8
SB (3)
SB (4)

【図４】


200 符号化データ
分離部
33
エントロピー復号器
32
CX
マルコフモデル生成部
31
参照順位テーブル
35
ST
状態生成部
34
画素インデックス（カラーシンボル）生成部
36
ラインバッファ
30
8
100B 画素インデックス（カラーシンボル）出力
変換カラーパレット18の出力

【図７】

(A)

R [3]	R [1]	R [2]
R [0]	buf [i]	

通常の参照画素

(B)

R [3] =0	R [1] =0	R [2] =0
R [0]	buf [i]	

先頭ラスタ

(C)

R [3] =0	R [1] =0	R [2] =0
R [0] =0	buf [i]	

先頭画素

【図８】

参照画素の状態	ST	色数	CX (マルコフ状態番号)
R [0] =R [1] =R [2] =R [3]	0	1	0
R [0] =R [1] =R [2] ≠R [3]	1	2	1
R [0] =R [1] =R [3] ≠R [2]	3	2	1
R [0] =R [2] =R [3] ≠R [1]	5	2	1
R [1] =R [2] =R [3] ≠R [0]	7	2	1
R [0] =R [1] ≠R [2] =R [3]	9	2	1
R [0] =R [2] ≠R [1] =R [3]	11	2	1
R [0] =R [3] ≠R [1] =R [2]	13	2	1
R [0] =R [1] R [0] ≠R [2] R [0] ≠R [3] R [2] ≠R [3]	16	3	2
R [0] =R [2] R [0] ≠R [1] R [0] ≠R [3] R [1] ≠R [3]	18	3	2
R [0] =R [3] R [0] ≠R [1] R [0] ≠R [2] R [1] ≠R [2]	21	3	2
R [1] =R [2] R [0] ≠R [1] R [1] ≠R [3] R [0] ≠R [3]	24	3	2
R [1] =R [3] R [0] ≠R [1] R [1] ≠R [2] R [0] ≠R [2]	27	3	2
R [2] =R [3] R [0] ≠R [1] R [1] ≠R [2] R [0] ≠R [2]	30	3	2
R [0] ≠R [1] R [0] ≠R [2] R [0] ≠R [3] R [2] ≠R [3] R [1] ≠R [2] R [2] ≠R [3]	33	4	3

【図９】

(A)

参照画素	符号化対象画素と一致した数
R [0] (=R [1] =R [3])	NA
R [2]	NB

(B)

NA≧NBのとき

参照画素位置	参照画素
0位	R [0]
1位	R [2]

(C)

NA＜NBのとき

参照画素位置	参照画素
0位	R [2]
1位	R [0]

【図１１】

画素インデックス (カラーシンボル)	参照画素中になかった度数
C_0	N_0
C_1	N_1
C_2	N_2
・	・
・	・
C_n	N_n
・	・
・	・
・	・
C_{255}	N_{256}

【図１０】

	参照画素位置			
ST	0位	1位	2位	3位
0	0			
1	0	3		
3	2	0		
5	1	0		
7	0	1		
9	0	2		
11	1	0		
13	0	1		
15	0	2	3	
18	1	0	3	
21	2	0	1	
24	1	0	3	
27	0	2	1	
30	1	2	0	
33	2	3	0	1

【図１２】

画素インデックス（カラーシンボル）	順位
C_0	1
C_1	⋮
C_2	0
⋮	⋮
C_n	255
⋮	⋮
C_{255}	⋮

【図１３】

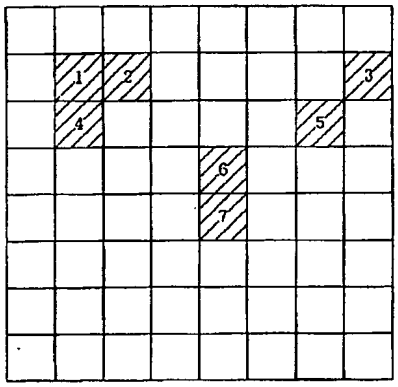
	入力されたカラーパレット
C_0	RGB_0
C_1	RGB_1
C_2	RGB_2
⋮	⋮
C_n	RGB_n
⋮	⋮
C_{255}	RGB_{255}

$C_2 \rightarrow C_0$
$C_0 \rightarrow C_1$
$C_n \rightarrow C_{255}$

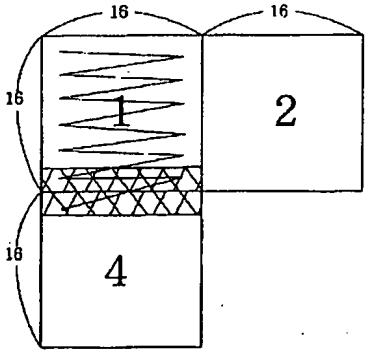
	変換されたカラーパレット
C_0	RGB_2
C_1	RGB_0
⋮	⋮
C_{255}	RGB_n

【図１６】

(A)



(B)

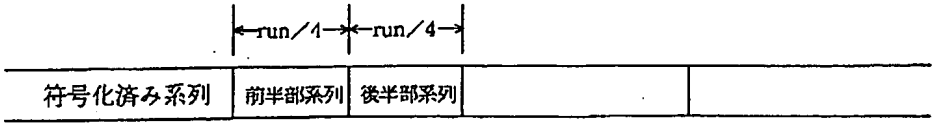


【図２４】

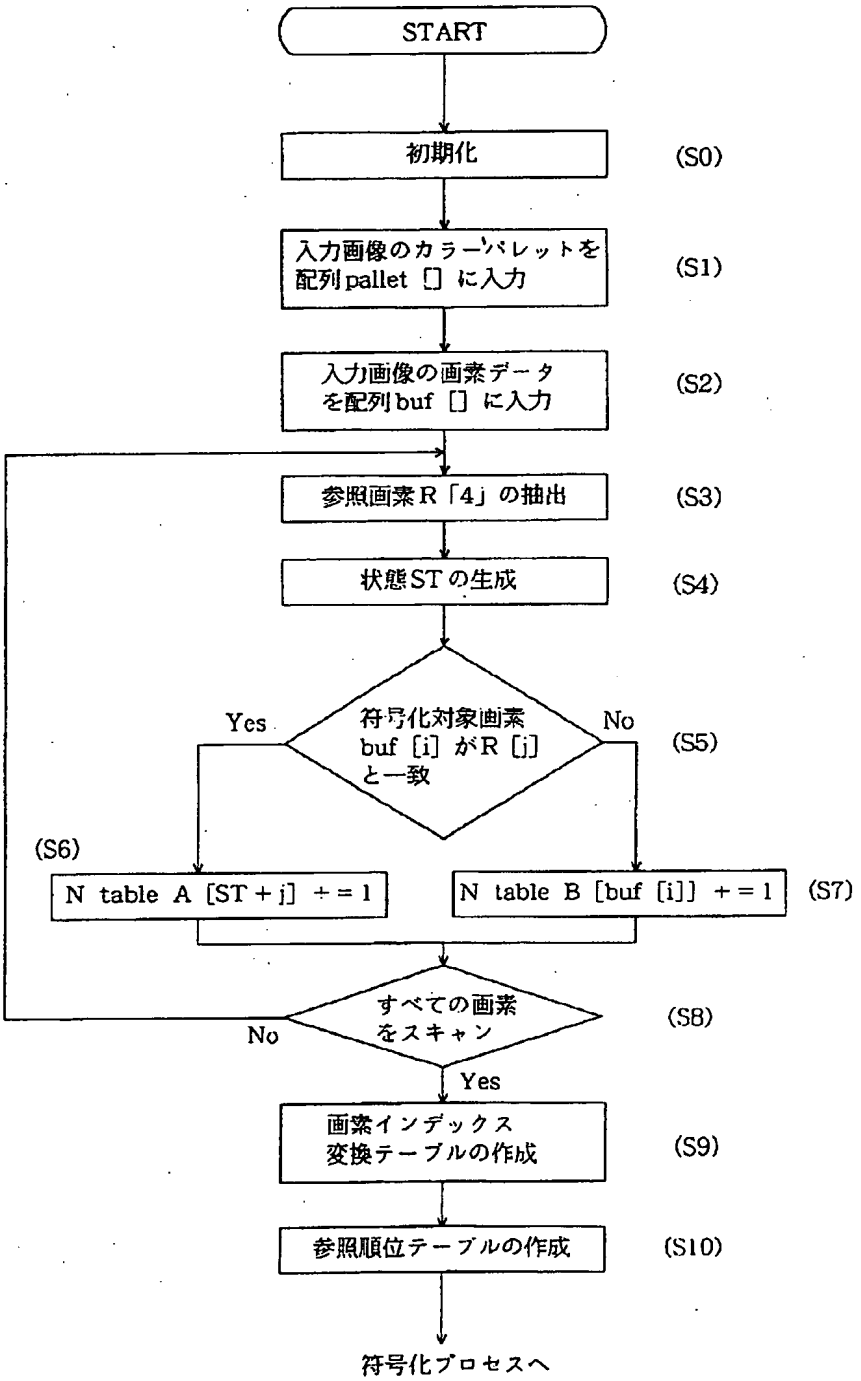
入力		出力		
DECNUM	DECPATN	DECBIT	LENGTH	FAIL
1	＊＊＊0	＊＊＊＊＊＊＊0	1	0
1	＊＊＊1	＊＊＊＊＊＊＊1	1	1

Table No.0 (run = 1)

【図２０】



【図１４】

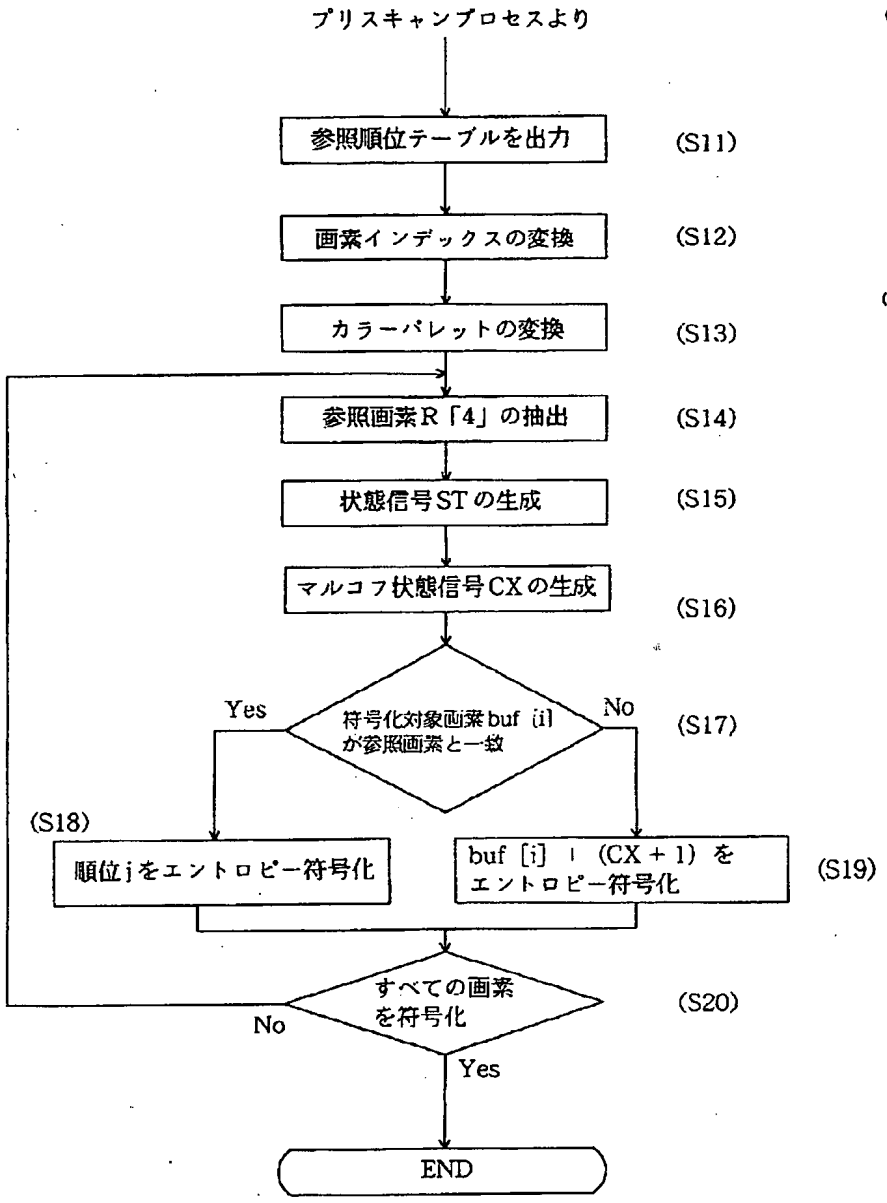


【図３０】

入力	出力			
CODE	LENGTH	DECNUM	DECPATN	FAIL
＊＊＊＊＊＊＊0	1	1	＊＊＊0	0
＊＊＊＊＊＊＊1	1	1	＊＊＊1	1

Table No.0 (run = 1)

【図１５】



【図３９】

(A)

C	B
A	X

X ：符号化対象画素
A,B,C：参照画素

(B)

条　件	Sx
A＝B＝C	S1
A＝B≠C	S2
A＝C≠B	S3
A≠B＝C	S4
A≠B≠C	S5

【図２５】

入力		出力		
DECNUM	DECPATN	DECBIT	LENGTH	FAIL
2	＊＊00	＊＊＊＊＊＊＊0	1	0
2	＊＊10	＊＊＊＊＊＊01	2	1
2	＊＊01	＊＊＊＊＊011	3	1
2	＊＊11	＊＊＊＊＊111	3	1

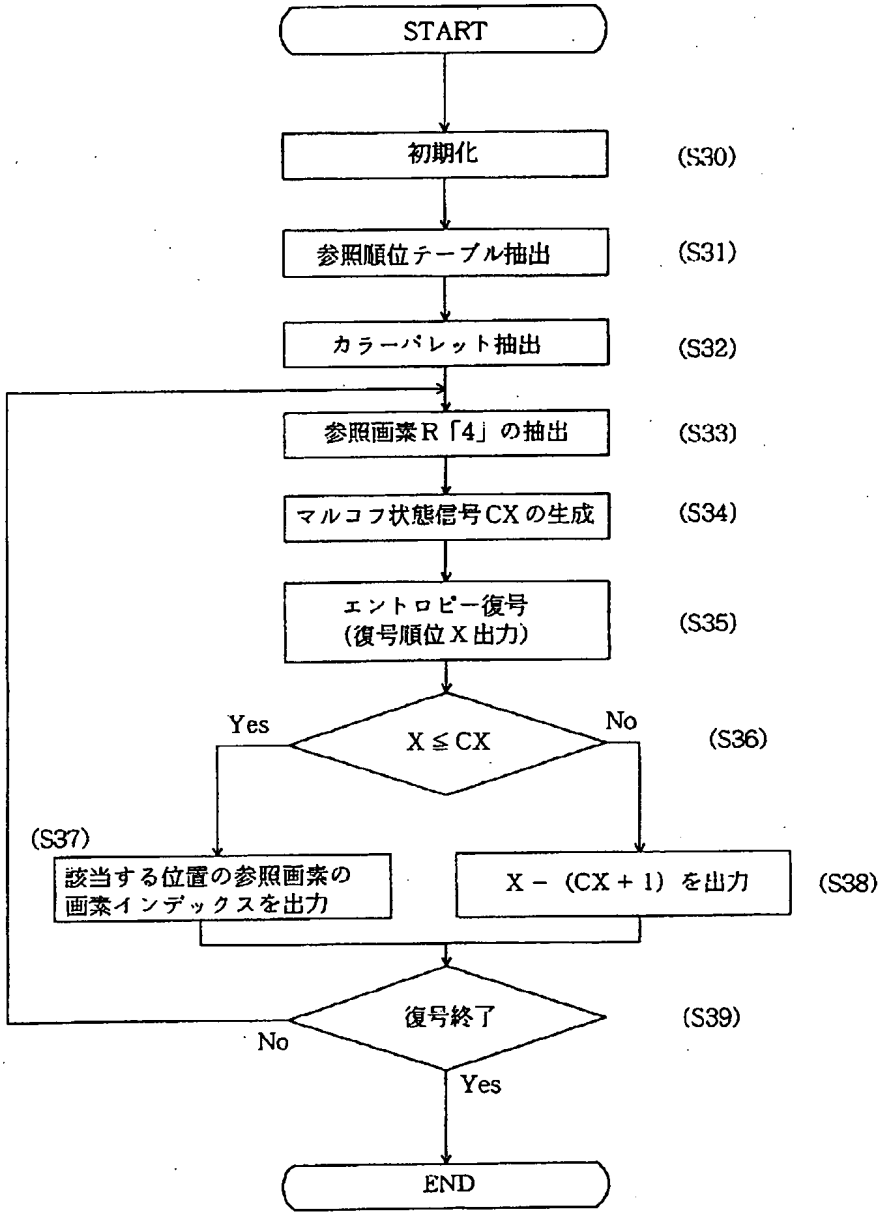
Table No.1 (run = 2)

【図３１】

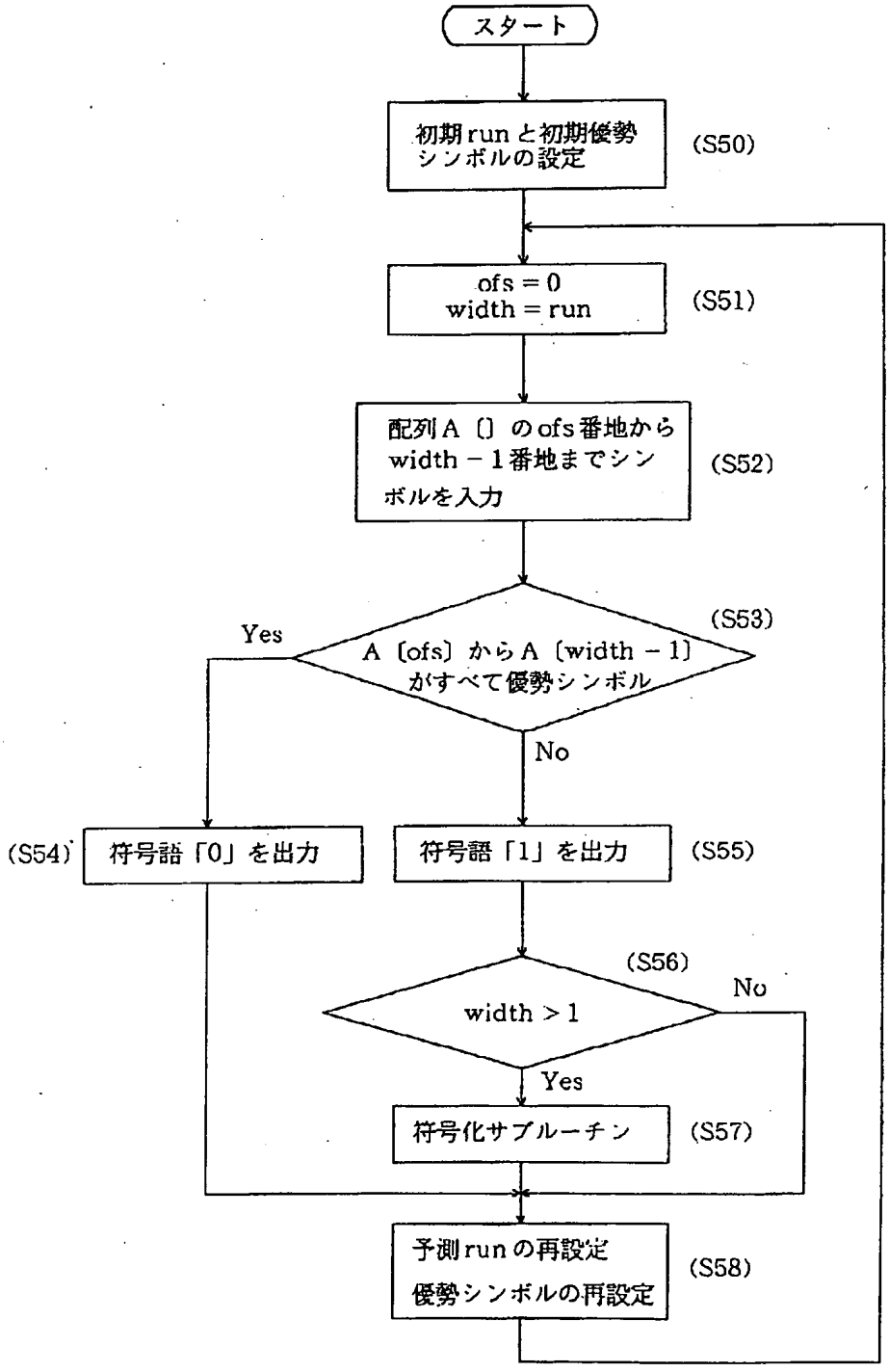
入力	出力			
CODE	LENGTH	DECNUM	DECPATN	FAIL
＊＊＊＊＊＊＊0	1	2	＊＊00	0
＊＊＊＊＊＊01	2	2	＊＊10	1
＊＊＊＊＊011	3	2	＊＊01	1
＊＊＊＊＊111	3	2	＊＊11	1

Table No.1 (run = 2)

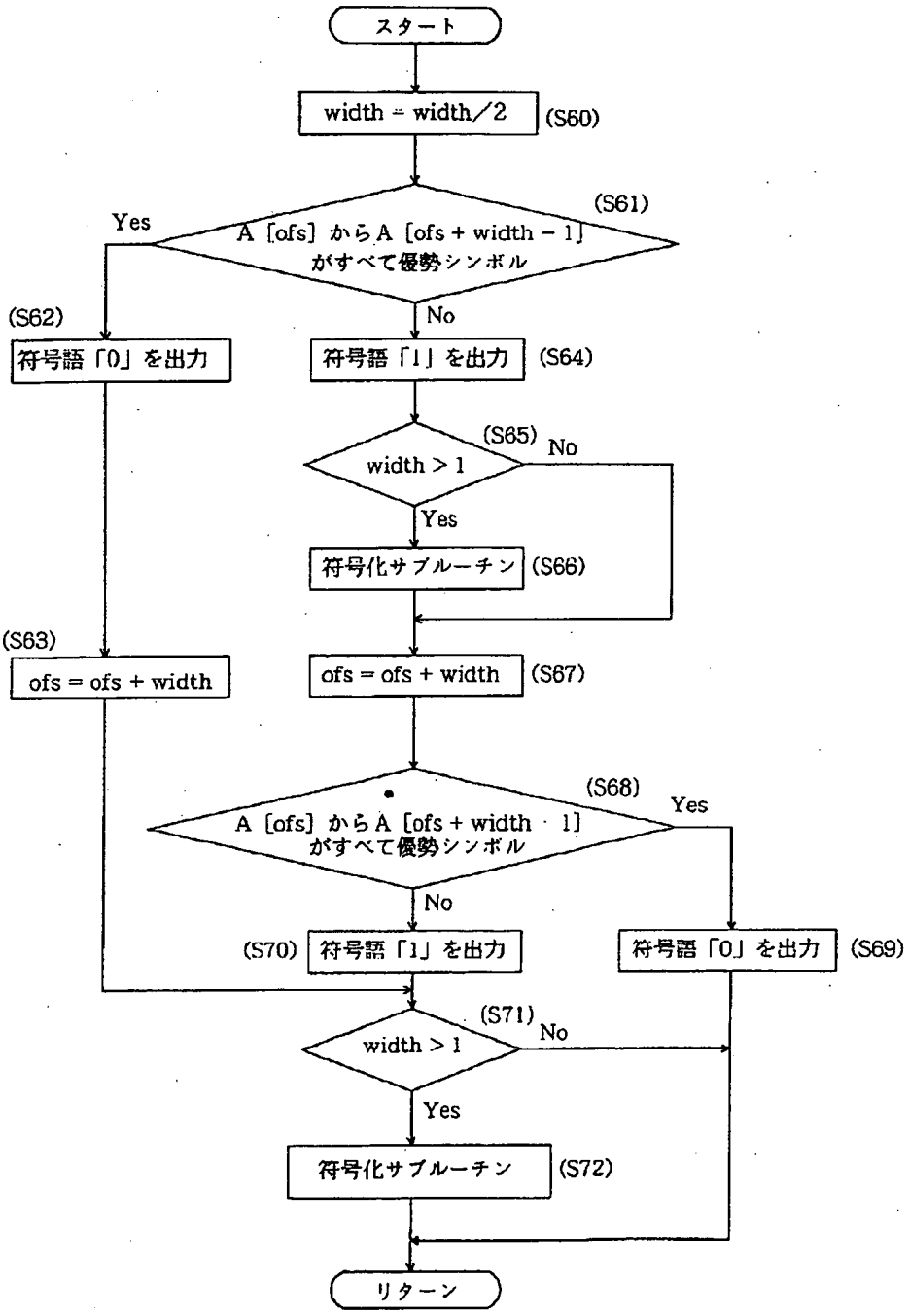
【図１７】


START
初期化
(S30)
参照順位テーブル抽出
(S31)
カラーパレット抽出
(S32)
参照画素R「4」の抽出
(S33)
マルコフ状態信号CXの生成
(S34)
エントロピー復号
(復号順位X出力)
(S35)
Yes
X≦CX
No
(S36)
(S37)
該当する位置の参照画素の
画素インデックスを出力
X－(CX＋1)を出力
(S38)
復号終了
(S39)
No
Yes
END

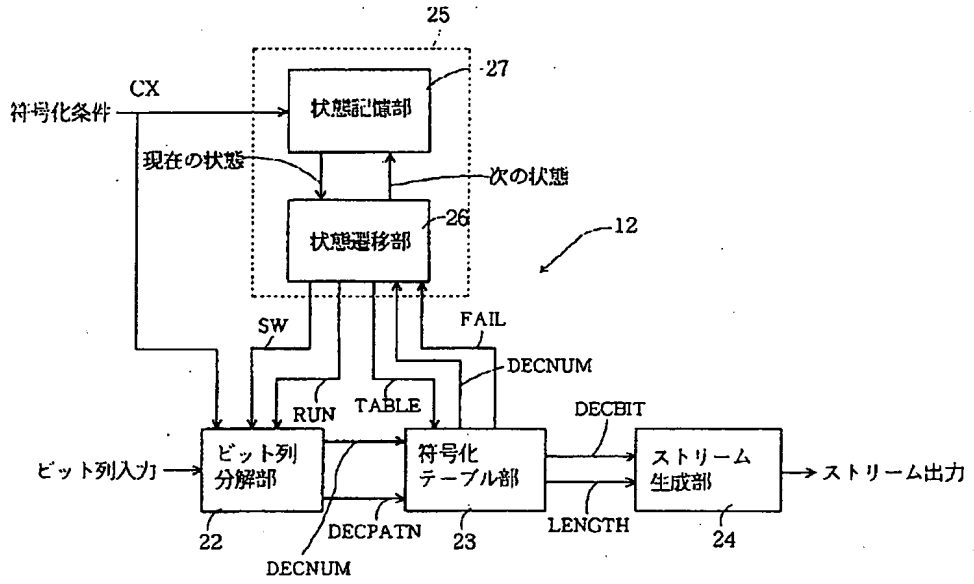
【図２１】


スタート
初期runと初期優勢シンボルの設定
(S50)
ofs = 0
width = run
(S51)
配列A〔〕のofs番地からwidth－1番地までシンボルを入力
(S52)
(S53)
Yes
A〔ofs〕からA〔width－1〕がすべて優勢シンボル
No
(S54)
符号語「0」を出力
符号語「1」を出力
(S55)
(S56)
No
width > 1
Yes
符号化サブルーチン
(S57)
予測runの再設定
優勢シンボルの再設定
(S58)

【図２２】


スタート
width = width／2
(S60)
(S61)
Yes
A [ofs] からA [ofs + width − 1]
がすべて優勢シンボル
No
(S62)
符号語「0」を出力
符号語「1」を出力
(S64)
(S65)
No
width > 1
Yes
符号化サブルーチン
(S66)
(S63)
ofs = ofs + width
ofs = ofs + width
(S67)
(S68)
Yes
A [ofs] からA [ofs + width − 1]
がすべて優勢シンボル
No
(S70)
符号語「1」を出力
符号語「0」を出力
(S69)
(S71)
No
width > 1
Yes
符号化サブルーチン
(S72)
リターン

【図２３】



【図２６】

入力		出力		
DECNUM	DECPATN	DECBIT	LENGTH	FAIL
4	0000	＊＊＊＊＊＊＊0	1	0
4	1000	＊＊＊＊＊001	3	1
4	0100	＊＊＊＊0101	4	1
4	1100	＊＊＊＊1101	4	1
4	0010	＊＊＊＊0011	4	1
4	1010	＊＊＊01011	5	1
4	0110	＊＊011011	6	1
4	1110	＊＊111011	6	1
4	0001	＊＊＊00111	5	1
4	1001	＊＊01011	6	1
4	0101	＊011011	7	1
4	1101	＊111011	7	1
4	0011	＊＊＊01111	5	1
4	1011	＊＊011111	6	1
4	0111	＊011111	7	1
4	1111	＊111111	7	1

Table No.2 (run = 4)

【図２７】

入力		出力		
DECNUM	DECPATN	DECBIT	LENGTH	FAIL
8	0000	＊＊＊＊＊＊＊0	1	0
8	1000	＊＊＊＊0001	4	1
8	0100	＊＊＊01001	5	1
8	1100	＊＊＊11001	5	1
8	0010	＊＊＊00101	5	1
8	1010	＊＊010101	6	1
8	0110	＊011010 1	7	1
8	1110	＊111010 1	7	1
8	0001	＊＊001101	6	1
8	1001	＊011011	7	1
8	0101	0111011	8	1
8	1101	1111011	8	1
8	0011	＊＊011101	6	1
8	1011	＊011110 1	7	1
8	0111	0111110 1	8	1
8	1111	1111110 1	8	1
4	1000	＊＊＊＊0011	4	1
4	0100	＊＊＊01011	5	1
4	1100	＊＊＊11011	5	1
4	0010	＊＊＊00111	5	1
4	1010	＊＊010111	6	1
4	0110	＊011011 1	7	1
4	1110	＊111011 1	7	1
4	0001	＊＊001111	6	1
4	1001	＊011011 1	7	1
4	0101	0111011 1	8	1
4	1101	1111011 1	8	1
4	0011	＊＊011111	6	1
4	1011	＊011111 1	7	1
4	0111	0111111 1	8	1
4	1111	1111111 1	8	1

Table No.3 (run = 8)

【図２８】

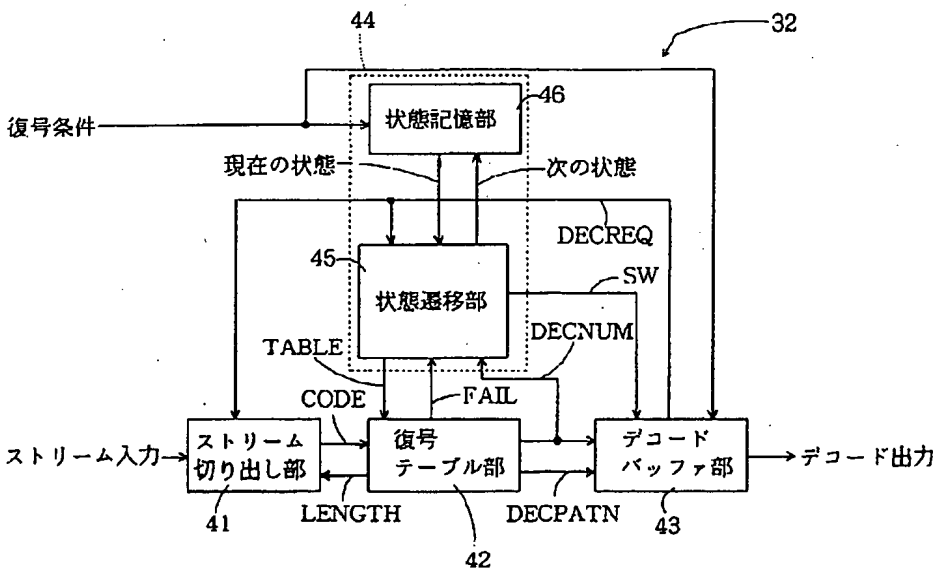
現状態		現条件であり、また次状態への出力		入力してくる結果		次状態準備	
現状態コード	現在の状態	run (RUN)	Table (TABLE)	FAIL	DECNUM	次の状態	SWの反転
0000	SS0	1	0	0	1	SS3	0
	SS0	1	0	1	1	SS1	1
0001	SS1	1	0	0	1	SS0	0
	SS1	1	0	1	1	SS1	1
0010	SS2	2	1	0	2	SS5	0
	SS2	2	1	1	2	SS3	0
0011	SS3	2	1	0	2	SS2	0
	SS3	2	1	1	2	SS0	0
0100	SS4	4	2	0	4	SS7	0
	SS4	4	2	1	4	SS5	0
0101	SS5	4	2	0	4	SS4	0
	SS6	4	2	1	4	SS2	0
0110	SS6	8	3	0	8	SS6	0
	SS8	8	3	1	8	SS7	0
	SS8	8	3	1	4	SS6	0
0111	SS7	8	3	0	8	SS8	0
	SS7	8	3	1	8	SS4	0
	SS7	8	3	1	4	SS5	0
1000	：	：	：	：	：	：	：
1001	：	：	：	：	：	：	：
1010	：	：	：	：	：	：	：
1011	：	：	：	：	：	：	：

【図３２】

入力	出力			
CODE	LENGTH	DECNUM	DECPATN	FAIL
＊＊＊＊＊＊＊0	1	4	0000	0
＊＊＊＊＊００１	3	4	1000	1
＊＊＊＊０１０１	4	4	0100	1
＊＊＊＊１１０１	4	4	1100	1
＊＊＊＊００１１	4	4	0010	1
＊＊＊０１０１１	5	4	1010	1
＊＊０１１０１１	6	4	0110	1
＊＊１１１０１１	6	4	1110	1
＊＊＊００１１１	5	4	0001	1
＊＊０１０１１１	6	4	1001	1
＊０１１０１１１	7	4	0101	1
＊１１１０１１１	7	4	1101	1
＊＊＊０１１１１	5	4	0011	1
＊＊０１１１１１	6	4	1011	1
＊０１１１１１１	7	4	0111	1
＊１１１１１１１	7	4	1111	1

Table No.2 (run = 4)

【図２９】



【図３３】

入力	出力			
CODE	LENGTH	DECNUM	DECPATN	FAIL
＊＊＊＊＊＊＊0	1	8	0000	0
＊＊＊＊0 0 0 1	4	8	1000	1
＊＊＊0 1 0 0 1	5	8	0100	1
＊＊＊1 1 0 0 1	5	8	1100	1
＊＊＊0 0 1 0 1	5	8	0010	1
＊＊0 1 0 1 0 1	6	8	1010	1
＊0 1 1 0 1 0 1	7	8	0110	1
＊1 1 1 0 1 0 1	7	8	1110	1
＊＊0 0 1 1 0 1	6	8	0001	1
＊0 1 0 1 1 0 1	7	8	1001	1
0 1 1 0 1 1 0 1	8	8	0101	1
1 1 1 0 1 1 0 1	8	8	1101	1
＊＊0 1 1 1 0 1	6	8	0011	1
＊0 1 1 1 1 0 1	7	8	1011	1
0 1 1 1 1 1 0 1	8	8	0111	1
1 1 1 1 1 1 0 1	8	8	1111	1
＊＊＊＊0 0 1 1	4	4	1000	1
＊＊＊0 1 0 1 1	5	4	0100	1
＊＊＊1 1 0 1 1	5	4	1100	1
＊＊＊0 0 1 1 1	5	4	0010	1
＊＊0 1 0 1 1 1	6	4	1010	1
＊0 1 1 0 1 1 1	7	4	0110	1
＊1 1 1 0 1 1 1	7	4	1110	1
＊＊0 0 1 1 1 1	6	4	0001	1
＊0 1 0 1 1 1 1	7	4	1001	1
0 1 1 0 1 1 1 1	8	4	0101	1
1 1 1 0 1 1 1 1	8	4	1101	1
＊＊0 1 1 1 1 1	6	4	0011	1
＊0 1 1 1 1 1 1	7	4	1011	1
0 1 1 1 1 1 1 1	8	4	0111	1
1 1 1 1 1 1 1 1	8	4	1111	1

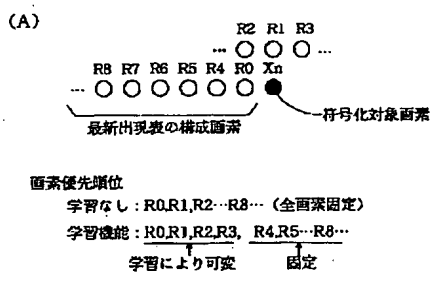
Table No.3 (run = 8)

【図３４】

グループ番号と付加ビット

入力シンボル	グループ番号	付加ビット数
0	0	0
1	1	0
2,3	2	1
4,5,6,7	3	2
8～15	4	3
16～31	5	4
32～63	6	5
64～127	7	6
127～255	8	7

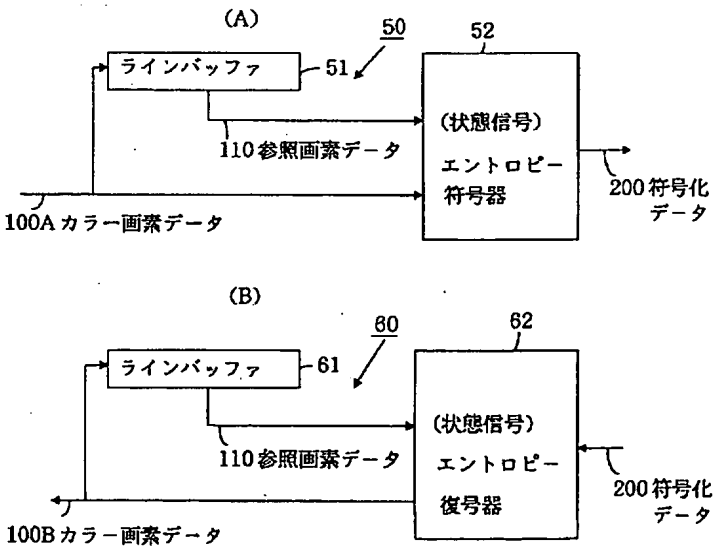
【図４２】



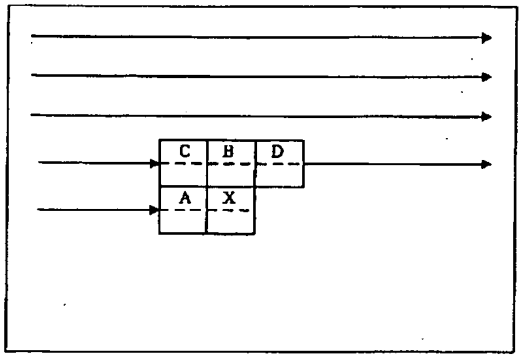
最新出現表

色順位	カラーシンボル
1位	C4
2位	C3
3位	C6
4位	C5
5位	C2
6位	C0
7位	C8
⋮	⋮
16位	Cn

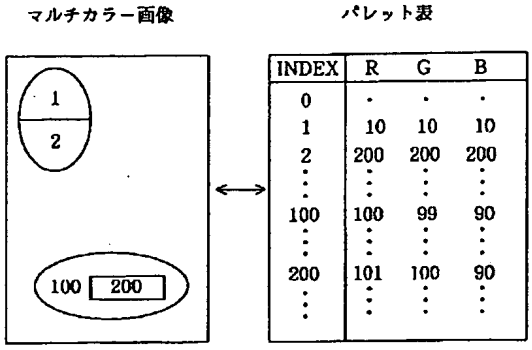
【図３５】



【図３６】



【図４１】

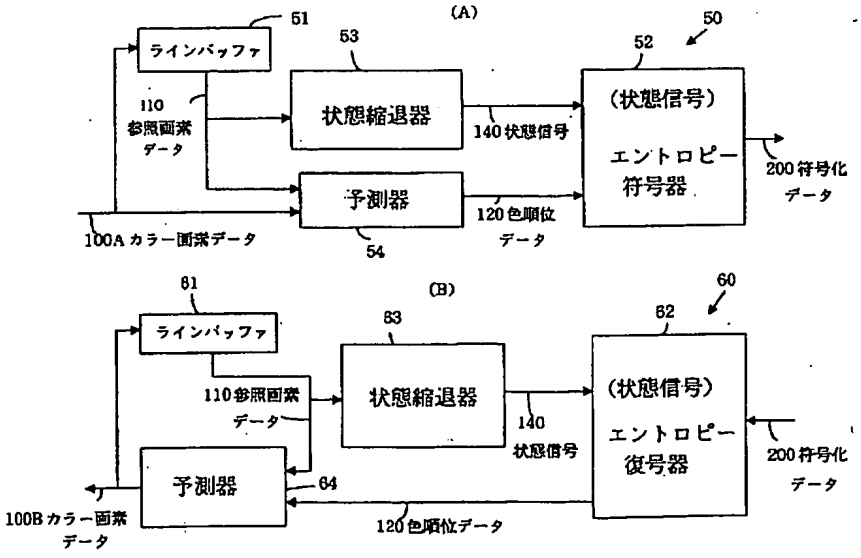


【図３７】

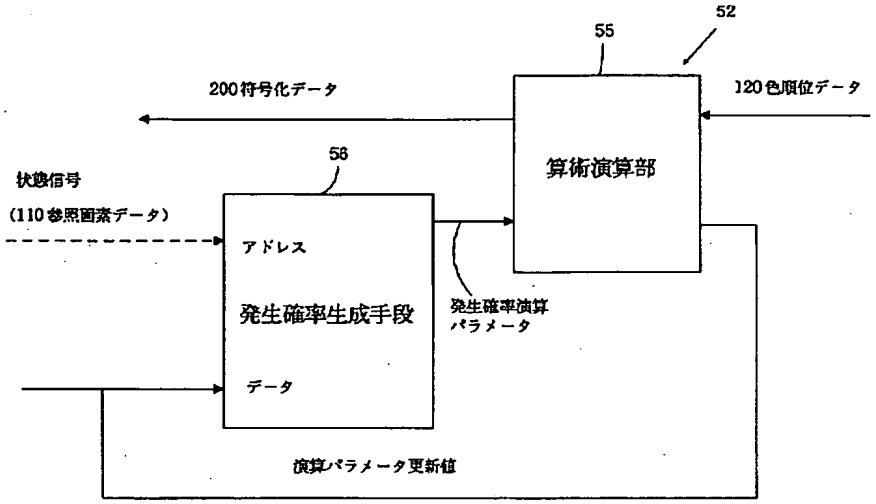
インデックス		各カラーシンボルの出現頻度						
		A	B	C	D	E		P
参照画素パターン	0	64,936	15,882	26,320	7,669	5,634		2,634
	1	5,553	36,214	18,776	20,365	6,046		953
	2	33,365	18,247	22,563	4,423	10,012		1,359
	3	9,569	33	13,456	562	356		224
	4	11,236	36,877	25,650	1,096	532		1,096
	65535	4,963	15,689	45,678	2,236	3,326		63,697

A,B,C ・・・・ P：カラーシンボル

【図３８】



【図４０】


52
55
200符号化データ
120色順位データ
算術演算部
56
状態信号
(110参照画素データ)
アドレス
発生確率生成手段
データ
発生確率演算
パラメータ
演算パラメータ更新値